THESE DE DOCTORAT DE L'UNIVERSITE PIERRE ET MARIE CURIE - PARIS VI

SPECIALITE INFORMATIQUE

Présentée par Ivan MIRO PANADES
Pour obtenir le titre de
DOCTEUR DE L'UNIVERSITE PARIS VI

CONCEPTION ET IMPLANTATION D'UN MICRO-RÉSEAU SUR PUCE AVEC GARANTIE DE SERVICE

Soutenue le 20 mai 2008 devant le jury composé de :

M. Jean-Marie CHESNEAUX	**(LIP6)**	**Président**
M. Giovanni DE MICHELI	**(EPFL)**	**Rapporteur**
M. Frédéric PETROT	**(INPG-TIMA)**	**Rapporteur**
M. Fabien CLERMIDY	**(CEA-Léti)**	**Examinateur**
M. Alain GREINER	**(LIP6)**	**Directeur de thèse**
M. Jean-Pierre SCHOELLKOPF	**(STMicroelectronics)**	**Co-directeur de thèse**

Remerciements

Je remercie tout d'abord le Professeur Alain Grenier pour son encadrement, son expérience et son soutien pendant ces trois années de thèse. Je le remercie également pour ses discussions techniques, ses conseils et sa participation aux corrections de cette thèse.

Je remercie vivement Abbas (Hamed) Sheibanyrad pour toutes ses idées et discussions techniques.

De même, j'exprime ma gratitude envers Jean-Pierre Schoellkopf pour m'avoir donné l'opportunité de faire une thèse industrielle au sein de STMicroelectronics à Crolles puis à Minatec.

Je remercie ensuite mes parents Magí et Isabel, mon frère Edgar, mes grand parents Agustí et Rosalia, ainsi qu'Elodie pour m'avoir permis de surmonter les moments difficiles de cette thèse.

Lors de ces trois années de thèse j'ai rencontré beaucoup de personnes qui ont contribué au bon déroulement de cette thèse. Je tiens à remercier Philippe Roche, Gilles Gasiot, Damien Giot, François Jaquet, Philippe Cavenel et Cyrille Dray pour m'avoir intégré au sein de l'équipe et avoir animé tous les repas de midi, Michel Harrand pour ses discussions techniques sur le NoC, Alain Tournier pour ses conseils sur la synthèse logique, Vincent Motel pour le support des outils de Cadence et Eric Vouriot pour le support de l'outil Encounter.

Je remercie aussi Thomas Finateu ainsi que Franck Badets pour m'avoir accordé leur confiance et pour avoir réussi à fabriquer un circuit fonctionnel, Fabrice Boissieres pour m'avoir formé aux outils de P&R, Pascale Maillet-Contoz et Shahin Mahmoodian pour leur efficacité, François Pécheux pour ses idées techniques et pour m'avoir accordé sa confiance au projet AOC.

Je tiens également à remercier Pascal Vivet, Fabien Clermidy, Edith Beigne, Yvain Thonnart et Didier Lattard pour m'avoir permis de récupérer la plate-forme FAUST. Grâce à eux j'ai pu aboutir à mon objectif final de thèse.

J'exprime ma sympathie à Alain Artieri pour son partage d'idées sur les circuits multiprocesseurs ainsi que pour m'avoir permis de participer à la définition d'une nouvelle architecture.

Enfin, je tiens à saluer tous mes amis, collègues et doctorants du Lip6, ST et Minatec avec qui j'ai partagé cette expérience.

Bonne lecture !

Résumé

Ce travail de thèse porte sur la conception et implantation physique d'un micro-réseau sur puce avec garantie de service. Ces études reposent sur le micro-réseau sur puce DSPIN développé au Lip6.

Dans un premier temps, nous étudions l'incorporation des communications avec garantie de service dans ce micro-réseau. Ce type de communications est très utilisé dans les systèmes ayant de fortes contraintes temporelles comme, par exemple, les traitements de flux vidéo ou audio. La solution proposée est capable d'offrir des garanties de latence et de bande passante à faible coût matériel.

Dans un deuxième temps, nous analysons une FIFO qui permet d'interconnecter des systèmes synchrones qui n'ont pas le même domaine d'horloge. Ce type de FIFO est optimisé pour des profondeurs faibles ainsi que pour faciliter son implantation dans des architectures compatibles avec l'approche Globalement Asynchrone, Localement Synchrone. Sa conception repose sur des cellules standard sans utiliser des cellules spécifiques ni asynchrones.

Enfin, nous présentons une implantation matérielle du micro-réseau DSPIN dans la plate-forme FAUST développée par le CEA-Léti. Toute la chaîne de conception, depuis la synthèse de l'architecture jusqu'au dessin des masques, est décrite en détail pour illustrer la façon dont la technologie DSPIN s'intègre dans un flot de conception industriel. Ainsi, le circuit final est testé avec des données réelles.

Mots-clés : Micro-réseau sur puce, implantation physique, garantie de service, NoC, DSPIN, FAUST, FIFO bi-synchrone, Globalement Asynchrone Localement Synchrone (GALS).

Titre en Anglais : Design and Implementation of a Network-on-Chip with Guaranteed Service.

Abstract

This dissertation addresses the design and the physical implementation of a Network-on-Chip (NoC) with guaranteed service traffic. In this context, the DSPIN Network-on-Chip developed at Lip6 was used as target architecture.

Firstly, we analyze the implementation of the guaranteed service traffic in the DSPIN architecture. This type of communications is largely used in real-time applications, such as video decoding or audio processing. The proposed solution features low area cost while delivering guaranteed latency and throughput.

Secondly, we analyze a bi-synchronous FIFO that is capable to interface synchronous systems working with different clock signals (frequency and/or phase). This FIFO is optimized for low depth while being compatible with the Globally Asynchronous, Locally Synchronous approach. Its implementation uses only standard cells, without either asynchronous or custom cells.

Finally, we present a physical implementation of the DSPIN architecture on the stream-oriented FAUST platform developed by CEA-Léti. The full implementation flow from synthesis up to mask design is detailed in order to illustrate the how the DSPIN technology is implemented on an industrial flow. As a final point, the circuit is verified with real data.

Key words: Network-on-Chip, physical implementation, guaranteed service, NoC, DSPIN, FAUST, bi-synchronous FIFO, Globally Asynchronous Locally Synchronous (GALS).

English Title: Design and Implementation of a Network-on-Chip with Guaranteed Service.

Table of Contents

Table of Figures

List of Tables

Introduction

Les avancées technologiques dans le domaine des circuits intégrés permettent d'introduire de plus en plus de transistors à l'intérieur d'une même surface de silicium. Dans les années 80, le nombre de transistors contenus dans un circuit intégré pouvait se compter par centaines. Désormais, les circuits intégrés les plus modernes contiennent plusieurs centaines de millions de transistors. Cette progression n'aurait pas été possible sans l'amélioration des procédés de fabrication et des outils de conception. Grâce à cette capacité d'intégration accrue, la réalisation de systèmes multiprocesseurs à l'intérieur d'un seul circuit intégré est désormais une réalité. Les avantages de ces type d'architectures sont multiples : grand nombre d'interconnexions (non limité par le nombre de plots du circuit intégré), communications intra-chip très rapides (par rapport aux communications inter-chip), réduction de la consommation (la capacité des condensateurs des plots est plus grande que celle des fils intra-chip), et coût de production réduit (l'intégration des composants dans un même circuit intégré réduit le prix total).

Jusqu'à présent, les interconnexions des systèmes inter-chip ont été conditionnées par le nombre de plots des circuits intégrés. De ce fait, les systèmes d'interconnexions de type bus ont été un succès car ils permettent de multiplexer sur les mêmes nappes de fils plusieurs communications. Les systèmes d'interconnexions intra-chip présentent deux problèmes majeurs. D'une part, les bus de données ont une limitation de bande passante car ils partagent la même ressource avec tous les utilisateurs. D'autre part, ces types d'interconnexions sont de plus en plus complexes à implanter physiquement car elles requièrent de très longs fils d'interconnexion, ce qui s'oppose à l'augmentation des fréquences d'horloge.

Les architectures de type micro-réseau sur puce (Network-on-Chip) offrent une bande passant largement supérieure aux bus étant donné que le système d'interconnexion est plus segmenté et que son nombre d'interconnexions est

supérieur. D'autre part, ils peuvent être conçus et implantés physiquement d'une manière plus simple que les bus d'interconnexion car les circuits intégrés peuvent être découpés en îles de communications indépendantes. Ce qui permet d'utiliser des techniques de conception de type Globalement Asynchrone, Localement Synchrone (GALS).

Le micro-réseau SPIN (Scalable Programmable Integrated Network) développé au LIP6, est la première architecture de micro-réseau à commutation de paquets à avoir été publiée. Elle visait à résoudre le problème du goulot d'étranglement constitué par le bus système dans les architectures multiprocesseurs à mémoire partagée intégrées sur puce (MPSoC). Par la suite, un grand nombre d'architectures de type Network-on-Chip (NoC) ont été publiées ÆTHEREAL, Nostrum, ANOC, Mango, QNoC, entre autres. En particulier, les concepteurs de ces micro-réseaux insistent sur la nécessité d'introduire des garanties de latence et de bande passante dans les communications pour des applications temps-réel.

Simultanément, l'implantation matérielle d'un micro-réseau SPIN à 32 ports chez STMicroelectronics a permis d'identifier les points faibles de cette architecture. Parmi les faiblesses de SPIN, nous trouvons que l'approche complètement synchrone n'est pas compatible avec les systèmes GALS, la topologie en arbre quaternaire élargi qui est peu modulaire, et finalement la réalisation par macro-cellule optimisée qui ne permet pas d'utiliser les bibliothèques de cellules précaractérisées fournies par le fondeur. Ceci nous conduit à définir l'architecture DSPIN (Distributed Programmable Integrated Network) capable de supporter des communications avec des garanties de service, synthétisable avec les cellules standard des fondeurs et compatible avec les architectures de type GALS.

Ce manuscrit s'articule de la manière suivant :

Le premier chapitre présente les trois principaux objectifs de notre travail : la conception d'un micro-réseau sur puce capable de supporter des communications avec des garanties de service, la conception des interfaces de communication entre systèmes qui ont des domaines d'horloge différents (phase et/ou fréquence) et les problèmes liés à l'implantation physique des micro-réseaux dans un flot de conception industriel. Simultanément, des questions ouvertes sont formulées pour mieux cibler les objectifs de ce manuscrit.

Le deuxième chapitre expose l'état de l'art pour la garantie de service et de l'implantation physique (l'état de l'art des interfaces de communication se trouve dans l'annexe A). Plusieurs architectures de micro-réseaux sont analysées en termes de communications avec garantie de service et de leur implantation physique. Au

début de ce chapitre, l'architecture SPIN est détaillée pour permettre de comprendre l'origine de DSPIN et son évolution vers une architecture distribuée.

L'introduction de la garantie de service dans l'architecture DSPIN est analysée dans le chapitre 3. Nous commençons par présenter une étude sur les garanties de service de type statistique avant de proposer une architecture utilisant deux canaux virtuels. L'architecture interne des modules de DSPIN (routeur et contrôleur d'interface réseau) est présentée de manière détaillée. L'allocation et l'acheminement des communications avec garantie de service sont analysées pour garantir une prédictibilité de ces communications en termes de latence et de bande passante. L'architecture DSPIN a été modélisée en SystemC et VHDL pour déterminer le seuil de saturation du réseau par simulation et confirmer les études analytiques sur la garantie de service. Finalement, l'architecture a été synthétisée, sa surface et sa fréquence maximale ont été caractérisées.

Le chapitre 4 expose la solution proposée pour interfacer deux systèmes synchrones qui ont des signaux d'horloge indépendants. L'objectif est de concevoir une FIFO de petite taille avec un coût de surface faible et synthétisable avec des cellules standard (sans utiliser de cellules asynchrones). Premièrement, un nouvel algorithme d'encodage est proposé et analysé. Cet encodage s'avère utile pour synchroniser des pointeurs de position entre deux systèmes d'horloge différents. Deuxièmement, cet encodage est utilisé dans la conception d'une FIFO de type bi-synchrone (deux interfaces synchrones contrôlées par deux horloges indépendantes). Nous présentons les schémas détaillés ainsi que diverses optimisations. Ces dernières permettent d'une part, d'améliorer l'utilisation de la FIFO et d'autre part, de réduire la latence de la FIFO quand les deux horloges, écriture et lecture ont la même fréquence mais une phase différente (mesochrone). La FIFO a été synthétisée en cellules standard pour caractériser sa surface et sa fréquence maximale en fonction du nombre de mots de la FIFO. Enfin, cette FIFO est comparée à d'autres FIFOs synthétisables.

Dans le chapitre 5, nous présentons une implantation matérielle du micro-réseau DSPIN dans la plate-forme FAUST développée par le CEA-Léti. Cette plate-forme contient plusieurs unités de calcul interconnectées par un micro-réseau ANOC. Ce micro-réseau a été remplacé par DSPIN (le détail de ce remplacement se trouve dans l'annexe B). Toute la chaîne de conception, depuis la synthèse de l'architecture jusqu'au dessin des masques, est décrite en détail pour illustrer la façon dont la technologie DSPIN s'intègre dans un flot de conception industriel. Une fois l'implantation conclue, le circuit a été caractérisé en prenant en compte les capacités

et résistances. Ainsi, le circuit final est testé avec des données réelles. Ce chapitre se conclut par une comparaison systématique entre les réseaux DSPIN et ANOC concernant la surface du micro-réseau, la bande passante, la latence des paquets pour traverser le micro-réseau, la puissance consommée et la manière d'être programmé.

Les conclusions de ce manuscrit sur les trois sujets abordés se trouvent dans le chapitre 6.

L'annexe A est un résumé de l'état de l'art des problèmes de synchronisation entre systèmes qui n'ont pas le même domaine d'horloge. Des solutions synthétisables ou semi-synthétisables sont analysées et comparées. Les architectures analysées sont des FIFOs avec contrôle de flux.

Les modifications introduites dans la plate-forme FAUST pour permettre l'intégration de DSPIN sont décrites en détail dans l'annexe B.

Enfin, l'estimation de la consommation des architectures DSPIN et ANOC dans la plate-forme FAUST, est détaillée dans l'annexe C.

Chapter 1

Problem Definition

This chapter introduces the problems addressed by this thesis. The scope of the analysis is limited to the Network-on-Chip domain. In the first section, the main problems addressed in this thesis are classified in three groups. For each group, some questions are formulated in order to be answered on the state of the art chapter, and on the proposed solutions.

A general introduction of the Network-on-Chip concepts is exposed in Section 1.2. In sections 1.3, 1.4, and 1.5, the problems elucidated on section 1.1 are detailed and analyzed.

1.1 Thesis motivation

Increasing the system performance by scaling the technology and the clock frequency becomes more and more complex due to the lower scalability of the wire delays. New approaches such as Network-on-Chip (NoC) architectures and the Globally Asynchronous, Locally Synchronous (GALS) paradigm tries to solve the design bottleneck by partitioning the circuit in small synchronous islands while they communicate asynchronously. Each island can be clocked by independent clock frequency, while the communications between neighbor islands are carried out by the NoC. Moreover, the NoC approach attempts to solve the bandwidth bottleneck of a central bus by splitting the communications over a plurality of routers and links.

On the other hand, the integration of many IPs into a single SoC requires handling a higher degree of predictability in terms of circuit performances. Thus, real time applications such as video, requires some sort of end-to-end guarantee traffic in order to achieve its required performances.

Finally, physical implementation of complex SoC with many IPs, memories, and a Network-on-Chip requires a simple and flexible implementation flow. Thus, a simple implementation complexity for the Network-on-Chip and the ability of partitioning the SoC simplifies the implementation time of complex circuits.

The contributions of this thesis can be grouped in three different topics, the quality of service in the NoC, the synchronization issues between independent clock domains, and the physical implementation of a NoC with independent clock domains. For each topic, some questions are formulated to focus the goals of this thesis. Detailed description of each topic can be found on the rest of the chapter.

1.1.1 Quality of Service

The introduction of the quality on service into a Network-on-Chip requires some sort of end-to-end path reservation in order to guarantee the latency and the throughput of the guaranteed service packets.

The questions addressed for this topic are:

- **Packet latency**: Which are the guarantees obtained on the packet latency? Are they hard bounded?
- **Throughput**: The throughput of the guaranteed service traffic is guaranteed? Is it hard bounded?
- **Overhead**: Which is the area/resources overhead of the NoC when the guaranteed service are introduced?
- **Shared resources**: What are the shared resources between best effort and guaranteed service traffic?
- **Path allocation**: How the guaranteed service traffic is allocated? By hardware or by software? Which is the complexity of this allocation?
- **GALS**: Is the NoC suited to the GALS approach?

1.1.2 Synchronization

The interface between two independent clock domains is vulnerable to a metastability failure. This topic analyzes the efficiency of the synchronization solutions on an NoC architecture.

- **Latency**: Which is the latency of the interface?
- **Throughput**: Is the interface able to deliver sustained 100% throughput? Which is the minimum FIFO depth to achieve 100% throughput?
- **Robustness**: Is the interface robust to the metastability failure?
- **Process, temperature, and voltage variation**: Is the interface robust to the process, temperature, and voltage variations?

- **Portability and industrialization**: Is the architecture suited to industrial implementation?
- **Testability**: Is the design test-friendly? Which type of test?
- **Density**: Which is the area of the interface?
- **Flexibility**: Is the physical implementation constrained by the design floorplan?

1.1.3 Physical Implementation

The physical implementation of a complex SoC requires increasing efforts on the Back-End flow. Thus, architectures that simplify the implementation flow are suited for multi-million transistor circuits. This topic is focused on the implementation complexity of the NoC, the GALS, and the mix-time interfaces.

- **Soft macro**: Is the NoC implemented as a hard or a soft macro?
- **Floorplanning**: Which are the physical and timing constraints of the NoC on the chip floorplanning?
- **Industrialization**: Is the NoC suited to be implemented on an industrial flow?
- **Portability**: Does the architecture contain asynchronous or custom cells?
- **Clocking**: How is implemented the clock-tree?
- **GALS**: Is the NoC suited to the GALS approach?
- **Clock boundaries**: How are implemented the clock boundaries?
- **Power**: Is the NoC efficient in terms of power?
- **Long wires**: How the long wires are implemented?
- **Predictability**: Is the NoC predictable, before and after Back-End?

1.2 Network-on-Chip Concepts

In this section, basic Network-on-Chip concepts are explained. The type of network, the network topology, the routing algorithm, the switching technique and the packet format are analyzed.

1.2.1 Packet Switching and Circuit Switching Networks

Two types of networks can be classified, the packet switching and the circuit switching networks. The former uses packets to communicate with the destination while the latter uses circuits. In the packet switching network, the packets contain de routing information needed to route the packet over the network. On the circuit-

switching network, an end-to-end circuit has to be established before any communication can happen. In a wormhole packet switching network, the packets are composed of flits. A flit is the smallest flow control unit handled by the network. The first flit of a packet is the head flit and the last flit is the tail. As soon as the head flit is received, the packet is routed to its destination. Moreover, the tail flit frees the router resources as soon it is routed. The main advantage and disadvantage between the packet switching and the circuit switching networks can be summarized in Table 1.1.

Table 1.1 Packet switching vs. circuit switching

Packet switching	Circuit switching
No circuit reservation	Need to allocate an end-to-end circuit
Packet contains the routing information (overhead)	No routing information as a circuit is established
Throughput depends on the network charge	Throughput is guaranteed once the circuit is allocated
Latency depends on the network charge	Lower and predictable latency once the circuit is allocated
Lower initial latency as no circuit has to be allocated	Higher initial latency as the circuit has to be allocated
Producer can send the information into the network even if the consumer is busy	Producer can only send the information if the whole end-to-end circuit is free. Otherwise has to retry later
Better network efficiency as when the packet is sent the resource is released automatically	The circuit have to be released to allow other to use it
Suited for best effort traffic	Suited for streaming and guaranteed service traffic
Require to decode each packet to route it	Require a circuit allocator to establish the circuit

1.2.2 NoC Building Blocks

The main blocks of a generic NoC are the routers, the network interface controllers, and the links. The routers are the switching units of the network, the network interface controllers behave as a bridge between the network and each local sub-system, and the links are the wires interconnecting them. Their principal task can be summarized:

- **Router**: Is the heart of the network. Its task is to route the packets over the network. Therefore, the packets are routed from a router input ports to the adequate router output port. The packets are not normally modified by the router; they are just forwarded to the adequate output port. However, some routing algorithms as the source-routing algorithm can modify the packet header on each router.

- **Links**: They are the wires interconnecting routers and connecting network interfaces to routers. These wires have to be properly buffered to guarantees a reliably communication in terms of crosstalk and noise immunity. Moreover, as technology size shrinks, the wire resistance and inductance becomes more important, and the wire latency becomes not negligible. Hence, more buffers are required to guarantee a reliable communication.
- **Network Interface Controller (NIC)**: Their main tasks are protocol conversion and packet building. The NIC provides services at the transport layer on the ISO-OSI reference model, offering to the local sub-system independency versus the network implementation.

1.2.3 Topology

The topology of a Network-on-Chip defines how the routers, links and network interface controllers are organized. The simplest topology is linear, where all the routers are connected inline as shown in Figure 1.1a. More complex topologies are ring, octagon, fat tree, 2D mesh topology, torus, and heterogeneous. Figure 1.1 shows some examples of regular topologies. The squares are the routing elements while the circles are the computing elements. The topology of the network conditions the implementation cost. The higher the number of connections (arity) per router, the higher the total bandwidth of the system, but also the higher the implementation cost of the router. Not all the topologies are good candidates for a silicon implementation of an NoC. The hypercube (n-cube when n>3), for example, is not suited as the two dimension nature of the actual circuits require long wire to implement a 3D structure i.e. increasing the implementation costs compared to a regular mesh topology. The selection of the topology of the network is a tradeoff between performance, complexity, and implementation cost. In [Bonon06] a comparison between the *ring*, the *mesh*, and the *spidergon* [Coppo04] topologies are analyzed in terms of diameter, scalability, and latency. A good choice for NoC is a regular topology with a simple routing algorithm.

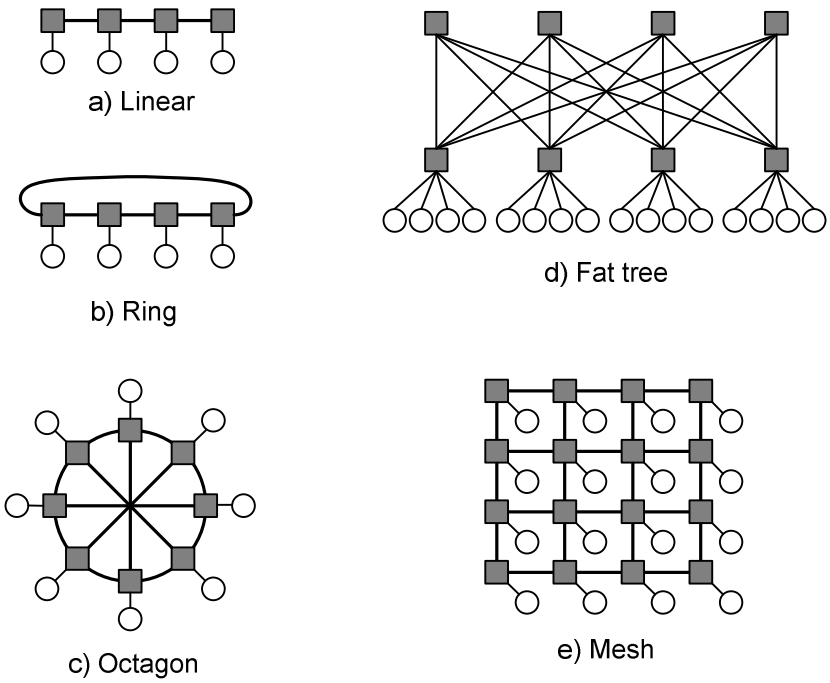


Figure 1.1 Regular topology examples

1.2.4 Routing Algorithm

In a packet switching network, the routing algorithm defines how a packet is routed to its destination. Two major types can be depicted: deterministic and adaptive algorithms. In deterministic routing, all the packets of a source-destination pair will follow the same path (in-order delivery). On the other hand, adaptive algorithms can modify the routing path in function of a metric (congestion, failure of a link, target busy …). Hence, the packets of a source-destination pair can follow different paths and the packets can arrive in a different order to that in which they were sent (out-of-order delivery). Thus, the target requires a reordering buffer to reorder the received packet. Some studies showed that the performances of the NoC can be improved using adaptive routing, but the silicon area of these reordering buffers should not be neglected. Adaptive algorithms can reduce hotspot (congestion) situations and/or avoid unreliable nodes or links. However, deterministic algorithms are the best choice for uniform or regular traffic patterns. Moreover, the global area of the system is optimized.

Even when an NoC is reliable, the routed packets can incur on failure situations. The most important causes of failure are:

- **Deadlock**: Two or more packets cannot reach its destination, because they are waiting for the other to finish. However, neither of them finishes.

- **Livelock**: A packet cannot reach its destination because it enters a cyclic path.
- **Starvation**: A packet cannot reach its destination because it does not have access to some resource while others have.

Deadlock and livelock are potential problems in wormhole [Bot04], [Dally01]. An analysis of these issues and how they can be solved is discussed below.

- **Deadlock**: Of the three issues summarized, deadlock is the most difficult to solve. It can be solved by two methods: deadlock prevention and deadlock recovery. The deadlock prevention is the most conservative. It guarantees no deadlock by construction, for example forbidding some turns on a 2D mesh topology as in the turn-model [Glass94]. Deadlock recovery techniques accept that in some situations the system can enter into deadlock situations. In that case, some special resources are used to break the dependences of the deadlock and recover a normal situation. Deadlock can also occur in a higher-level communication paradigm, for example, the request/response packets of a shared memory system can lead into deadlock situations if they are not treated as dependent traffic (a request and its response have a dependency). Basic solutions consist of splitting the request and the response into independent virtual channels or by splitting the network into two independent sub-networks, one for the requests and the other for the responses).
- **Livelock**: The system enters in a livelock due to routing loops on the routing algorithm or due to adaptive algorithm as the deflection routing (also known as hot potato routing). These issues can be probabilistically avoided [Duato03] or circumvented by using minimal routing path.
- **Starvation**: It is the simplest issue to solve. It is the consequence of unfair allocation policies of the resources. Its solution is to use a fair allocation policy.

Popular deterministic deadlock-free algorithms on a 2D mesh topology are X-first, Y-first, West-first, and Negative-first. All of them follow the premises of the turn-model to avoid the deadlock situations. The X-first algorithm, first routes the packets through the X direction until it reaches the corresponding X coordinate and then routes through the Y direction until its destination. The Y-first is similar to the X-first, but it first starts by the Y coordinate and then through the X. The West-first is similar to the last two but the packet is first sent to the west side. The Negative-first

has the particularity to send the packet first to a lower X coordinate or to a lower Y coordinate in function of the destination.

1.2.5 Switching Algorithm

The switching algorithm determines how the packets are forwarded between switches. The switches have input and output ports which can contain FIFOs. Those FIFOs are temporary storage for the packets. If the FIFOs are placed on the input ports, it is an input queuing switch, and if they are placed on the output ports, it is an output queuing switch.

The most popular packet switching algorithms are:

- **Store-and-forward**: In this switching algorithm, the packets are forwarded to the next router only when the whole packet has been received. It means that the routers must contain enough buffering space to receive the longest packet. Moreover, the latency of the packets depends on the length of the packets, as the packets are not forwarded until the end of packet is received.
- **Virtual cut-through**: It is similar to the store-and-forward but the packet can be forwarded to the next router as soon as the packet header is received, limiting the packet latency and the required buffering space.
- **Wormhole**: It is similar to the virtual cut-through but the packet is decomposed into trailing packets (flits), thus reducing the buffering space.

Wormhole switching algorithm has lower latency and requires smaller buffering space than the others require. However, as a packet may occupy many intermediate switches at the same time, livelock and deadlock situations occur more often in wormhole than for the others. For the same reason, the network congestion (two packets try to access the same output port) is increased in wormhole switching algorithm as a stalled packet can congest many routers. An example of input queuing switch congestion is depicted in Figure 1.2. This phenomenon is amplified when the network speed is higher than the speed of the computing elements. In that case, the packet advances faster over the routers than the computing element could generate it. Thus, the packets became elongate, they occupy many routers, and they generate congestion on many routers.

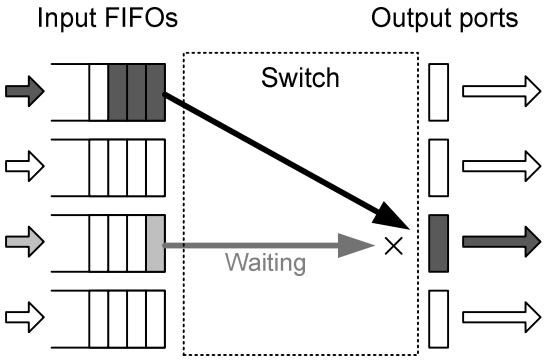


Figure 1.2 Congestion on an input-queuing switch

Virtual channel is a way to multiplex independent communications over the same physical links [Dally87]. This switching technique can be implemented on circuit switching or packet switching architectures. On circuit switching, the virtual channel can create virtual circuits by multiplexing the circuits on the links. The number of virtual channels that can be supported by the link depends on the number of buffers of the link [Miche06]. Two schemas have been used: a buffer per virtual channel (spatial distribution) or a buffer per link (temporal distribution).

1.2.5.1 Virtual channel with a buffer per channel

The virtual channels are spatially distributed on the switch thank to independent buffers. The channels are temporally multiplexed over the same link using time slots. The allocation policy of the physical link can be static (as round-robin) or dynamic. Static allocation guarantees a reserved bandwidth per virtual channel (example: 1 of N time slots is reserved to a virtual channel i) while dynamic allocation can maximize the allocation of the link (example: a virtual channel is allocated when there is data to transfer and there is enough space on destination buffer). Figure 1.3 shows an example where four virtual channels are multiplexed.

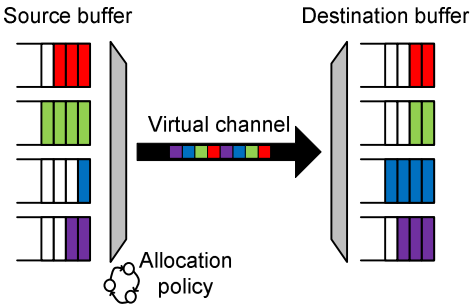


Figure 1.3 Virtual channels with a buffer per channel

1.2.5.2 Virtual channel with a buffer per link

It is possible to use just a buffer per link when the switching allocation and the virtual channel data are temporally multiplexed and synchronized. This is normally achieved by statically scheduling the data and the switching allocation. Thus, many virtual channels can share the same input port. Figure 1.4 shows an example where many virtual channels share the same input port. Independent traffic data has different color.

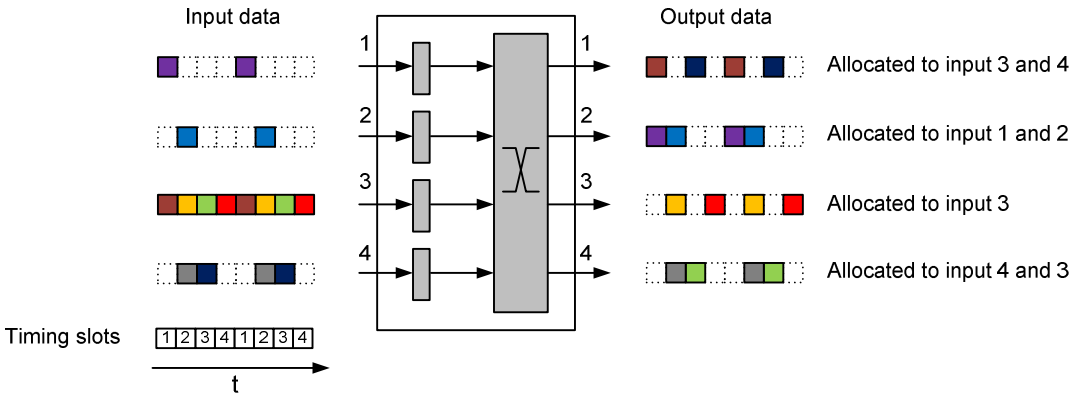


Figure 1.4 Virtual channel with a buffer per link

When the system is correctly synchronized and all the virtual traffics are temporally disjoin, there is no need to have local flow control mechanism. Therefore, an end-to-end flow control mechanism can be employed as a circuit-switching network.

This methodology requires low buffering space and less multiplexors compared to the virtual channel with a buffer per channel. However, it requires a global synchronicity of the network and a global slot allocation of the virtual channels in order to archive high performances.

1.2.6 Header Encoding

As seen in the routing algorithm section, the packet carries the routing information to allow the router to forward the packets to the right direction. On a wormhole switching router, this information is stored in the first flit of the packet. Thus, the routing decision can be taken as soon as the first flit of the packet is received. The routing information can be the absolute address (address-based) of the destination or the routing path (source routing algorithm). The former, uses a fixed number of bits to encode all the possible absolute addresses in the network. The latter, uses a fixed number of bits to describe the successive hops between routers to

reach the destination. Figure 1.5 shows a possible header format for these algorithms. An analysis of their features is discussed below.

- **Address-based:** On a mesh topology, the absolute address can be defined using N bits (Nx + Ny = Nbits), where Nx and Ny are the number of bits to encode the maximum X and Y coordinate respectively. The clear separation of the Nx and Ny bits allow a rapid decoding of the X and Y addresses. Therefore, a compact implementation of the routing algorithm is possible. For a network architecture containing 256 units (16,16), it is possible to define the addresses using 4bits on Nx and 4bits for Ny. Hence, the routing information occupies 8bits. Another less advantageous distribution could be (5, 52) topology which means 5 * 52 = 260 units. In this later case Nx requires 3 bits and Ny requires 6 bits, consequently N is 9 bits. The length of the routing information grows logarithmically with the system size; on a 10x10 system, N requires 10 bits.

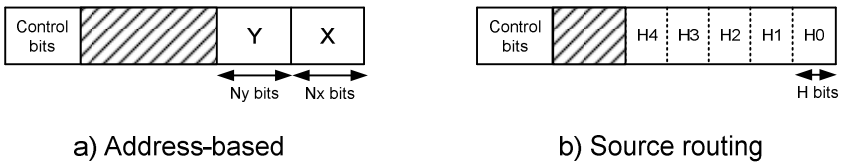


Figure 1.5 First flit definition

- **Source routing:** The packet contains the routing path. Its size depends on the maximum number of hops (H0 to Hn) and on the number of bits to encode each hop (H bits) as shown in Figure 1.5b. At each hop, the router has to decide using H bits to witch output port sends the packet. Once the hop is done, the routing information is shifted H bits. The H bits encode one of the possible output ports of the router. Thus, the router can use a look up table to determine the proper output port to route the packet. On a mesh topology where the routers have 5 ports (north, south, east, west, and local), H can be defined using 3 bits (000-North, 001-South, 010-East, 011-West, and 100-Local). Alternatively, it can be defined using 2 bits in a more intelligent way; it is the case of ANOC NoC [Beigne05]. In this NoC, it is established that a packet cannot be routed to the same direction as it came i.e. a packet coming from west cannot be routed to the west port. Using this premise, the code to route the packet to the local port is the incoming port as shown in Figure 1.6. Assuming a mesh topology of Nx columns of Ny rows of routers (Nx,Ny), the minimum number of hops to

reach any destination is defined using the Manhattan distance of the two opposite vertices of the mesh (0,0) to (Nx-1,Ny-1). Its distance is (Nx-1) + (Ny-1) + 1 = Nx+Ny-1 hops. Assuming Nx=Ny=5, the required number of hops is 9. Taking into account that each hop requires 2 bits, the routing path needs at least 2*9=18 bits. If the system contains (10,10) units, the routing path requires at least 19*2=38 bits. It is clear that the routing path needs more bits than the address-based when the number of routers increases (38 bits compared to 10 bits).

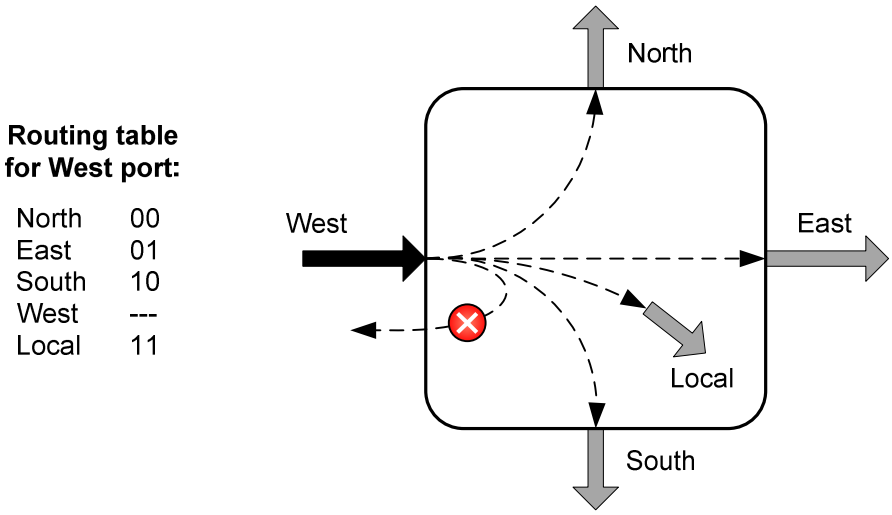


Figure 1.6 Source routing of ANOC NoC

Table 1.2 summarizes the advantage and disadvantages of these two routing algorithms.

The main drawback of the *Source routing* is the lower scalability compared to the *Address-based*. When the routing bits do not fit into a flit size, some path extension has to be used to override this limitation. In the FAUST implementation using the ANOC NoC [Beigne05], two path-extensions modules were added to overcome this limitation.

In order to use the path-extension modules, the routing path is encoded in the first and second flits of the packet. When the packet passes through a path-extension module, the first flit of the packet is erased. Thus, the packet is routed using the routing information on the second flit, which now is the first flit. With this technique, it is possible to extend the routing path as long as the packet length. Nevertheless, the system architect has to avoid hot spot situations into these path-extension modules when many packets run out of routing path.

Table 1.2 Advantage and disadvantage of absolute address and routing path

Address-based	Source routing
Small overhead on the packet. Requires 10 bits on a 10 * 10 system	Higher overhead. Requires at least 38 bits on a 10*10 system.
A 32bit address can define a 65536 * 65536 system. It does not require address extensions for the current circuits.	If the routing path excesses the flit data size (example 32bits) some routing extensions have to be done. Otherwise, with a flit size of 32bits the maximum system size is 8*9 = 72 units.
Requires arithmetic operations on the router to decide the routing path.	Fast decision on the router. Needs a look up table of 2^H entries.
The producer of the packet does not need to know the routing path.	Each producer has to know the routing path to its destinations to build the routing path.
Better implementation for generic applications and multiprocessor systems.	Better implementation on specific applications with fixed data flows. As each packet producer requires the routing path of its destination.
Better implementation for homogenous systems with regular topologies.	Better implementation for heterogeneous and non-regular topologies.
The routing algorithm on the routers manages the congestion of the links.	The packet producer can deroute the packets to avoid congested links.

In terms of programming model, *Address-based* is optimal for shared-memory architectures where the MSB bits of the address can be used to define the (X,Y) coordinated on a mesh topology. On the other hand, *source routing* requires knowing all the possible destinations to translate the address into the routing path. Therefore, a programmable Look up table is required.

1.2.7 Router Functionalities

The routers are the switching modules of the network. They are composed of input and output ports, FIFOs, multiplexers, and state-machines. An analysis of the routing steps on a wormhole packet switching, input queuing, and deterministic routing is discussed below:

- The first flit of a packet arrives to the router through an input port.
- The flits are stored into a FIFO.
- The first flit of the packet is decoded using the routing algorithm to find the corresponding output port, and a request is sent to this output port.
- The output port includes a state-machine that receives the request from the input ports. It allocates the output port to the chosen communication through a multiplexer. If several requests arrive at the same time, it arbitrates the requests to produce a fair allocation of the output port.
- The flits starts to flow from the input FIFO to the output port.
- The output port remains allocated until the end of the packet.

- The output port is desallocated when it receives the end of the packet. Thus, the output port can be used by other communications

1.2.8 Network Interface Controller

The network interface controller (NIC) is the only access to the network for a local subsystem. Usually, its architecture follows the ISO-OSI 7498 [ISO] reference model and it brings the *transport* layer of the network. This protocol layer guarantees the routing of the packet to its destination. Thereby, it hides all the communication/synchronization issues to the local subsystem. The NIC main tasks are protocol conversion and packet building. The protocol conversion allows a clear separation between computation and communication. At the producer side, it builds packets using the routing information, the flit flow-control bits and the data to be sent. At the consumer side, it restores the information as if the consumer was directly connected to the producer. More sophisticated network interfaces can manage error detection and correction, packet retransmission, interrupt handling signals, and multiple data flows.

1.3 Quality of Service

Some applications as the video and audio decoding require a constant and guaranteed data flow between the pair producer-consumer. For these applications, the traffic flow between the pair producer-consumer should have some guarantees on the latency, on the throughput, and on the latency jitter. Traditional packet switching networks do not offer these types of guarantees as all packets share the same resources. The quality of service (QoS) refers to a resource reservation mechanism where special packets do not share the resources with other packets. The special packets are called Guaranteed Service (GS) packets while the others are called Best Effort (BE) packets.

Best Effort is the basic service of a network and it does not support any kind of QoS. Networks that just support BE service will try to satisfy all the communications at the same time. Therefore, the latency of the packets cannot be bounded because the network capacity can be exceeded. Figure 1.7 shows the probability distribution of the latency for a 10*10 network with different FIFO depths. Even when the FIFO is deep (22 words), the latency of the packet cannot be bounded. These kinds of networks are well suited for general-purpose applications without well-defined data-flows.

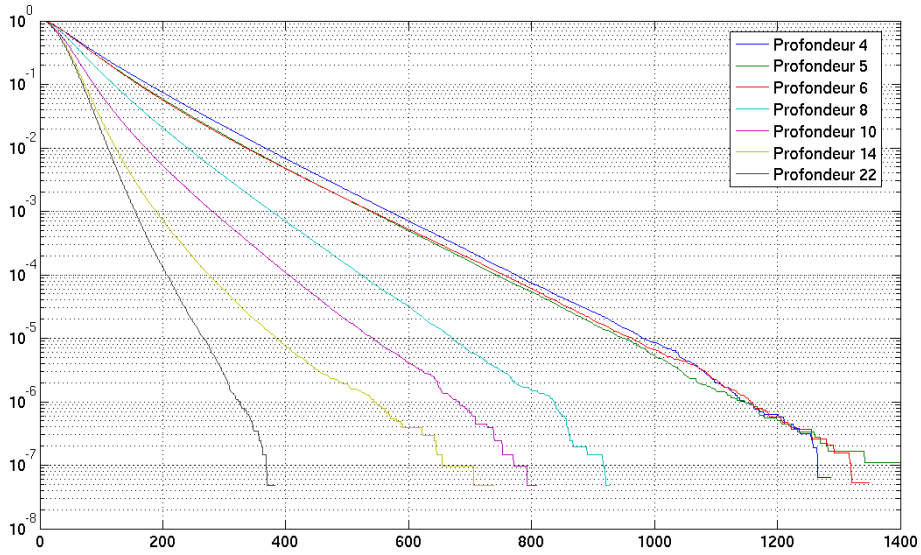


Figure 1.7 Porbability distribution of the packet latency (clock cycles) in function of the FIFO depth

The introduction of the QoS has been largely studied in wide-area networks; however, two major differences exist between on-chip and off-chip networks:

- **Dropping packets**: Off-chip networks can drop packets due to buffer overflows, misrouting or router failure. The buffer overflow situation comes from the fact that the channels interconnecting the off-chip routers are very long and the packets are deeply pipelined. Packet dropping can be avoided in on-chip networks thanks to smaller inter-router wire delay and the utilization of flow control mechanisms.
- **Cost elements**: The most expensive element on the off-chip networks are the inter-router wires, while buffering memory is less expensive. Thus, to minimize the network cost, the number of wires per link will be reduced to the maximum while the less expensive elements will be dimensioned to optimize the wire utilization. On the on-chip networks, the memory area is tightly related to the circuit cost, while the number of wires between routers is not the costly element. Thus, the shared resources, wires and FIFOs, are treated in a different manner as the tradeoff between cost/performance depends on the type of network.

The network bandwidth is bounded. Therefore, it is not always possible to satisfy all GS communications. A traffic contract has to be established for each GS traffic to avoid exceeding the network capacity. This contract establishes the maximum throughput, the maximum latency, the maximum jitter, and the duration of the transmission for each pair producer-consumer requiring a QoS. These parameters are

used by a network entity to allocate the required network resources to satisfy the GS request. This network entity can be a centralized processor or a distributed network allocator.

In terms of type of QoS, we can define two major types of guarantees:

- **Hard** is the strictest type of guarantee. It guarantees the maximum predictability of the network. The latency, throughput and jitter are bounded and well constrained. This kind of guarantees can be achieved granting the exclusiveness of some resources to the guaranteed traffic.
- **Soft** is a less strict than the hard one. It guarantees some metrics (latency, throughput, and jitter) but it has some degree of unpredictability. This kind of guarantees can be achieved by mixing some exclusive and nonexclusive resources.

1.4 Synchronization Issues

In deep sub-micron processes, the largest parts of the delays are related to the wires. The ITRS [ITRS05] report details the evolution of the wire delay in function of the process node as shown on Figure 1.8. In multi-million gates System-on-Chip (SoC), achieving timing closure is difficult, as place & route tools have difficulty coping with long wires and balancing the clock tree.

The Globally Asynchronous, Locally Synchronous (GALS) [Chapiro84] [Mutter99] approach attempts to solve this problem by partitioning the SoC into isolated synchronous islands that have frequency and phase clock independency. With this approach, the timing constraints of the SoC can be bounded to the isochronous limit of each island. In this case, the communications between islands should be carried out by mixed-timing interfaces that adapt the clock frequency and phase discrepancy. Such interfaces are not trivial [Ginosar03] since the synchronization failure (metastability) of the registers can corrupt the transferred data. The main issues on these mixed-timing interfaces and how to prevent it is discussed below.

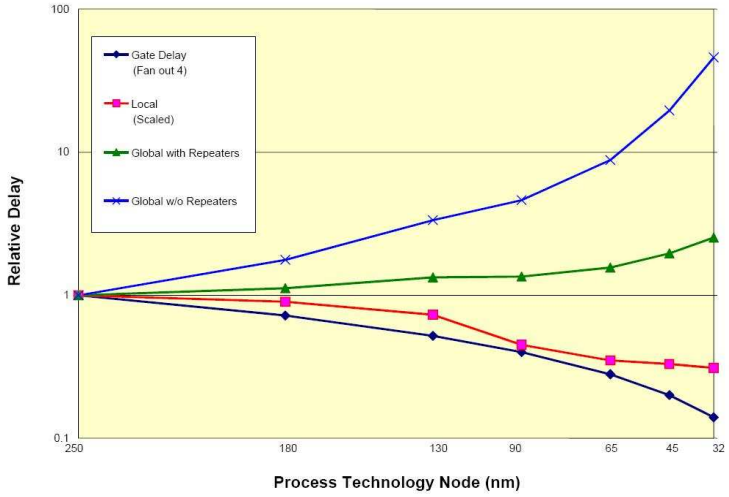


Figure 1.8 Delay of metal1 and global wiring versus feature size [ITRS05]

1.4.1 Metastability

Metastability is the ability of a non-equilibrium state to persist for a long, theoretically unbounded, time. In electronics, this phenomenon can appear on the flip-flops since they are designed to have two logic states (1 and 0). However, between these two stable states it is possible to identify a metastable state as seen in Figure 1.9. It is possible to set the flip-flop on the metastable state if the input data changes between the setup-time and hold-time.

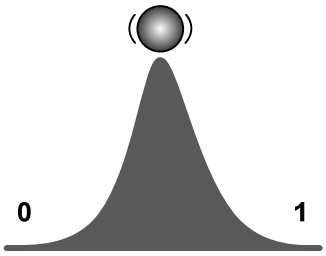


Figure 1.9 Metastable state

1.4.2 Metastability on Cross-Coupled Inverters

Most flip-flops contain cross-coupled inverters that have the ability to retain data. These cross-coupled inverters constitute the memorizing capability of the flip-flop, which decides the logical state of the flip-flop. Figure 1.10 shows a cross-coupled inverters and the evolution of the output in function of the input signal. The output of a cross-coupled inverter tries to force a defined logic level 0 or 1. When the input signal is High the output is Low and vice versa. The feedback loop boosts the input port enforcing the cross-coupled inverts to define a clear output state 0 or 1. Hence, the input signal needs to be stronger than the top coupled

inverter; otherwise the output state will not change. In some circuits, the top inverter of Figure 1.10 is weaker than the bottom one. Therefore, the required input signal strength to switch the output can be lower.

The cross-coupled inverters have a defined stable states of logic 0 and logic 1 states (Vss and Vdd electric levels respectively). Furthermore, the output of the cross-coupled inverters has a metastable state when the input = output = Vdd/2.

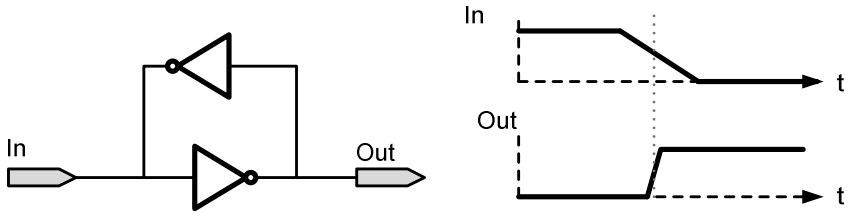


Figure 1.10 Cross-coupled inverter and its output signal in function of its input

When the inverters are not balanced, the conductance of the N transistor is higher/lower than the conductance of the P transistor, the metastable point is modified but not eliminated (Figure 1.11) [Tamir03]. The metastable point can be founded intersecting the red and the blue curves. These curves are the inverter transfer function of the top and bottom inverters.

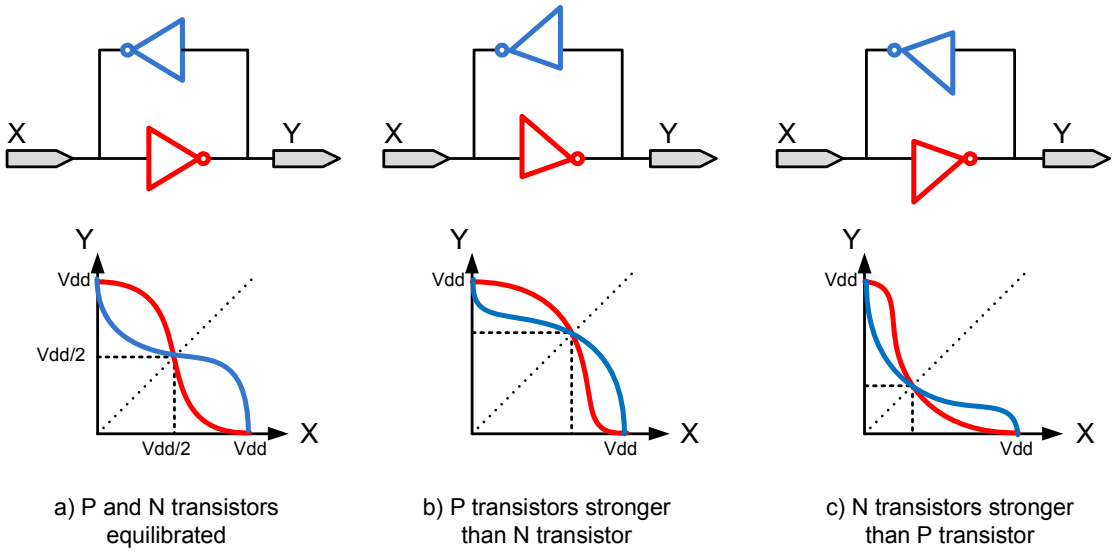


Figure 1.11 Metastable point in function of the transistor aspect ratio

The resolution time is the time it takes the cross-coupled inverter to leave the metastable state. The resolution time is minimized when the gain-bandwidth product of the cross-coupled inverters is maximized. Lee-Sup Kim and Robert W. Dutton analyze the metastability of the CMOS latches and flip-flops in [Kim90].

1.4.3 Metastability on Flip-Flops

Once defined the issues on the cross-coupled inverter, an analysis of the metastability issues on the flip-flop is discussed. Figure 1.12 shows a simplified architecture of a D flip-flop. This kind on flip-flop is massively used in digital synchronous circuits. The input data (D) is latched on the rising edge of the clock (CK) signal.

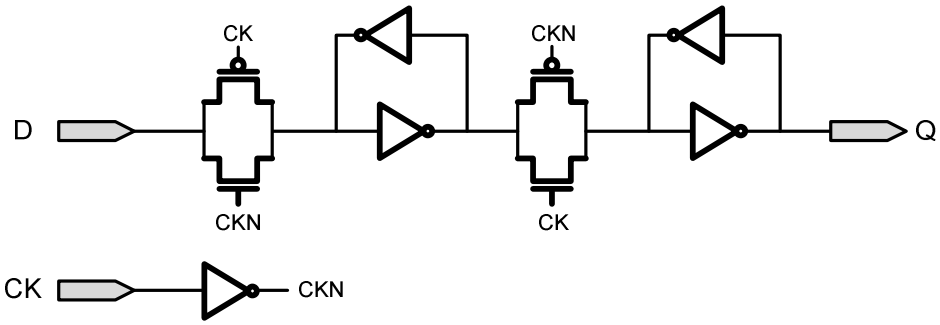


Figure 1.12 Simplified view of a D flip-flop

If the input signal D is Vdd/2 when the clock rising edge arrives, the flip-flop can became metastable. To prevent these situations, the setup-time and the hold-time of the flip-flop have to be respected. The time window defined by the setup-time and the hold-time of the flip-flop defines a time interval where the flip-flop response time is not guaranteed. If the input changes inside this time frame, the flip-flop will require more time to output a valid data. Figure 1.13 shows the degradation of the access time when the setup-time is violated. The yellow line is the input clock signal, the blue lines are the input data for different input delays, and the green lines are the obtained output data. The output delay of the flip-flop is perturbed due to a setup violation. The closest to the metastable point, the higher the output delay.

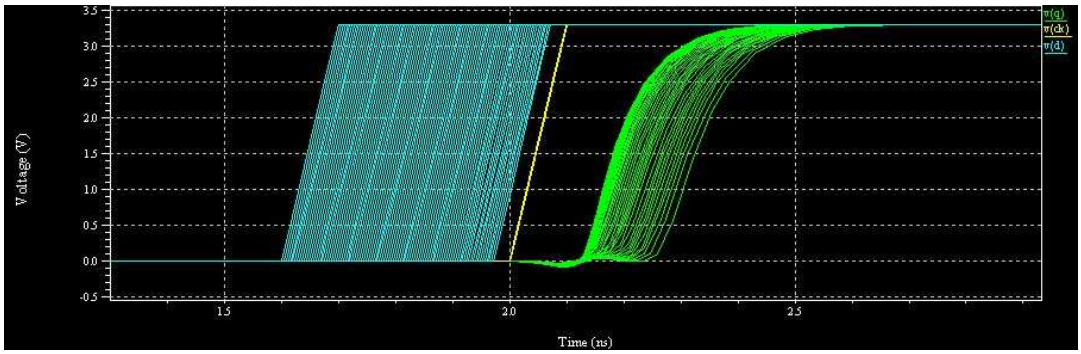


Figure 1.13 Delay degradation after setup-time violation [Tamir03]

Figure 1.14 shows the D flip-flop output for different input delays. These delays have been chosen to illustrate the output waveform of the D flip-flop when it is near the metastable point. The output waveforms are finally resolved to Vss or Vdd. The longest output delay is obtained when the D flip-flop is near the metastable point. In this situation, the thermal noise can help to decide if the output data will be resolved to Vdd or Vss as the flip-flop is in an instable point.

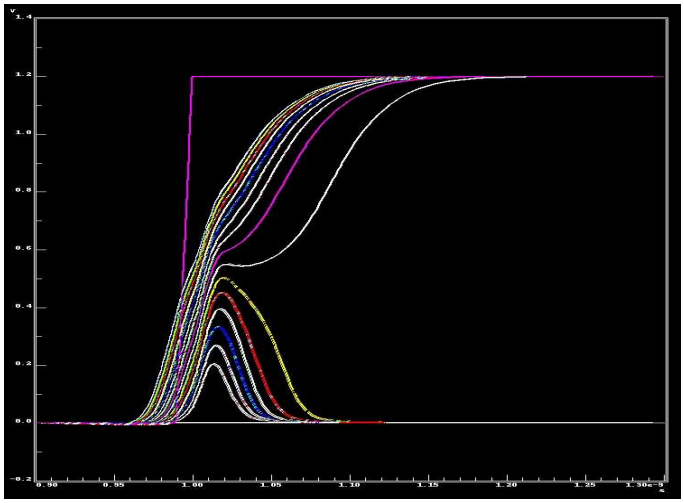


Figure 1.14 D flip flop output near the metastable point

1.4.4 Metastability Resolution and Robustness

The output delay of the flip-flop near the metastable point has been analyzed by Charles Dike and Eduard Burton on [Dike99] and represented by the histogram on Figure 1.15.

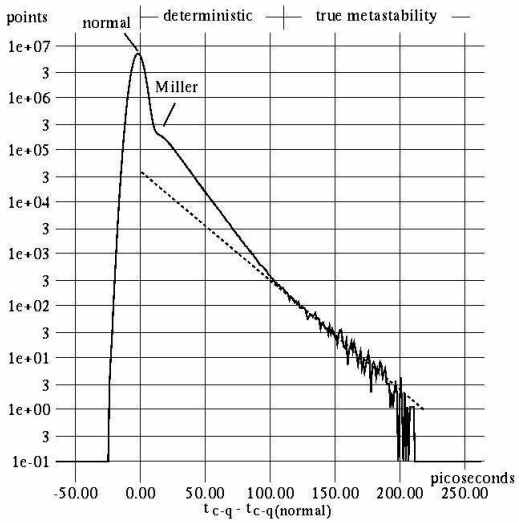


Figure 1.15 Resolution time histogram

It is not possible to define a hard bound for the resolution time but a Mean Time Between Failure (MTBF) can be calculated with Equation (1) where:

- **T**: Required resolution time.
- **τ:** Settling time of the flip-flop.
- **W**: Effective size in picoseconds of the metastability window at a normal propagation delay.
- **F**C: Clock frequency.
- **F**D: Frequency of data edges capable of generating metastability.

$$MTBF = \frac{e^{\frac{T}{\tau}}}{W * F_c * F_D} \quad \textbf{(1)}$$

This equation calculates the theoretic time between two failures of the flip-flop in function of some design parameters. The *settling time* of the flip-flop depends on the internal architecture of the flip-flop. The *required resolution time* is the maximum time we allow the flip-flop to decide to a stable logic value. The longer the *required resolution time*, the higher the MTBF. Likewise, the lower the operating frequency, the higher the MTBF.

A simple and safe method to maximize the MTBF is the two-flop synchronizer [Dike99][Kinni02][Dally98] as depicted in Figure 1.16. In this architecture, the first flip-flop samples the asynchronous data and resolves the metastability. The second flip-flop waits a full clock period before latching the synchronized data. Thereby, the intermediate signal X can take up to one clock cycle to stabilize before being latched. With this architecture, the *required resolution time* is maximized to one clock cycle without modifying the sampling clock frequency. In some special situations where the obtained MTBF is not robust enough, a three-flop synchronizer can be used. Thereby, the required resolution time is elongated to two clock cycles.

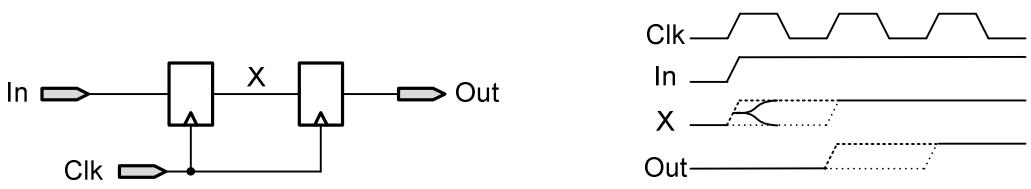


Figure 1.16 Two-flop synchronizer

1.4.5 Common Errors

This section is principally taken out from [Ginosar03] a reference paper for the designer on multi-synchronous interfaces. Here, some common errors are discussed and analyzed to identify the potential metastability issues on the synchronization of synchronous systems.

1.4.5.1 One-flop synchronizer

This architecture is a simplification of the two-flop synchronizer. The one-flop synchronizer eliminates the second flip-flop of the two-flop synchronizer. As seen in previous paragraph, the *required resolution time* of a two-flop synchronizer is one clock period. With a one-flop synchronizer, the *required resolution time* is no longer a full clock period as the combinational logic has some latency. Consequently, the resolution time of the one-flop synchronizer is reduced, because now there is some combinational logic between the first flip-flop and the next flip-flop. With one-flop synchronizer, the *required resolution times* is reduced to the slack (T-C) between the one-flop synchronizer and the next flip-flop. Hence, the MTBF decreases.

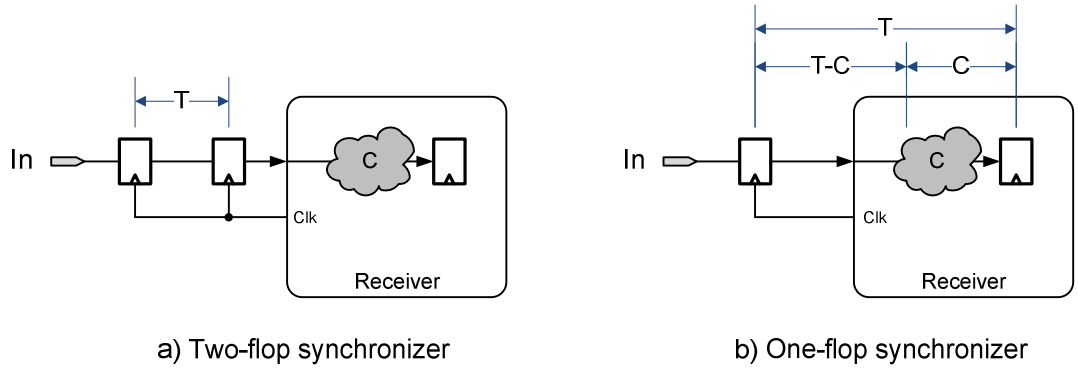


Figure 1.17 Two-flop and one-flop synchronizers

1.4.5.2 Parallel synchronizer

This architecture tries to synchronize the data of a multi-bit word by using a two-flop synchronizer per bit. Figure 1.18 shows a representation of this architecture. This scheme seams to be a correct synchronizer as all the data lines are synchronized and no combinational logic are inserted between the flip-flops. However, the synchronized data will not always be correct. If the input data changes near the metastable window, each two-flop synchronizer can end up doing something different: some can take the data before the rising edge, others can take the new data, and other may enter in metastable state and settle to 0 or 1. There is no way to

guarantee the coherency of the multi-bit data at the output. A coherent multi-bit data cannot be synchronized using a parallel synchronizer.

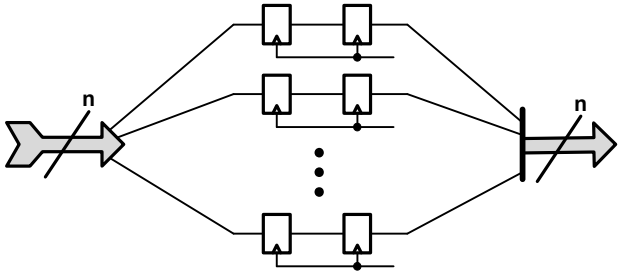


Figure 1.18 Parallel synchronizer

1.4.5.3 Global reset synchronizer

Most of the circuits share the same global reset signal to initialize the global circuit. However, as the circuit uses multi-frequency domains, the initialization cannot be successfully accomplished due to metastable situations of the registers. Firstly, the rising and falling edges of the reset signal have to be considered on a different manner because the critical issue on the reset signal is the falling edge, not the rising. At the reset rising edge all the flip-flops are forced to initialize its contents. The reset signal can remain active many clock cycles to guarantee the correct initialization of the whole system. Asynchronous-reset flip-flops initialize at the rising edge of the reset signal while the synchronous-reset flip-flops initialize when the reset signal is high and the rising edge of the clock signal arrives. Finally, the falling edge of the reset releases the system to its normal operation. If the falling edge of the reset signal comes near the rising edge of the clock signal, the flip-flop can lose the input data or became metastable. To avoid this situation, the release of the reset signal has to be synchronized with the local clock signal. Figure 1.19 shows a safe interface for the reset signal. The rising edge is directly propagated to the system while the falling edge is synchronized with a two-flop synchronizer. This system has two features: the falling-edge of the reset is well synchronized with the clock signal and the rising-edge of the reset is propagated asynchronously to the system through the OR gate. The latter feature allows activate the reset signal asynchronously though a huge system and disperses the dropping power of the initialization over a time window. Otherwise, if thousands of flip-flops have to be initialized at the rising edge of the clock, the circuit power lines may not be enough large to bring the required power consumption.

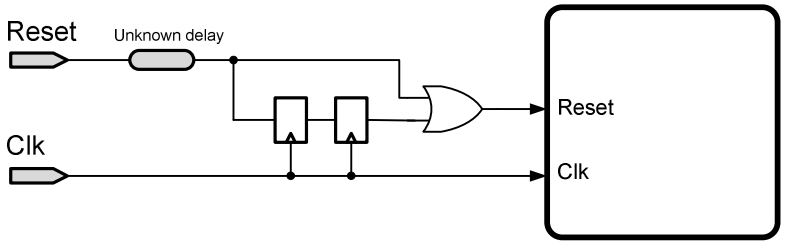


Figure 1.19 Global reset synchronizer

1.4.6 Clock Relationship

The clock relationship between two independent synchronous domains can help to simplify the interface. This relationship can be in terms of frequency and/or phase. The basic clock relationships between two systems are summarized below.

- **Synchronous**: Both systems have the same clock frequency and phase. No special assumption has to be made to interface them.
- **Mesochronous**: Both systems receive the same clock signal but a difference of phase (skew) exists. This skew is constant over the time. This is the case of an unbalanced clock tree distribution over a circuit.
- **Plesiochronous**: Both systems receive a similar clock frequency but it is not possible to guarantee a constant difference of phase between them. This is the case where each clock signal is generated by independent oscillators or PLL tuned to the same clock frequency.
- **Rational**: The clock frequency signal of one system is rational value of the clock frequency of the other system. In this case is possible to predict the rising edges of the clock signals and interface the system.
- **Asynchronous**: There is not relationship between the clock signals of both systems. The clock signal is generated by independent sources.

For each of these clock relationships, optimized solutions that minimize the latency and the area have been proposed.

1.5 Physical Implementation Complexity

In this section, some basic concepts related to the physical implementation are exposed. The Front-End and Back-End flows are introduced where its mains steps are detailed. Finally, the long wire delay issues are and the circuit power consumption are analyzed.

1.5.1 Hard, Firm and Soft Macros

Macro is the abbreviation for a virtual component or IP core. These macros are designed to be reused and ported to different applications or process technologies. So, they are classified in terms of the degree of optimization for a particular fabrication process:

- **Hard macros** are optimized for power, size or performance when mapped to specific chip technology, and usually delivered in GDS-II format. Being process-specific, hard macros have the advantage of having deterministic timing, area, and power-consumption characteristics. Its drawback is that they are process-specific and they are not portable to other process technology. Nevertheless, the circuit architecture is more protected than the soft and firm macros. The SRAM memories and processors such as the ARM [ARM] are examples of *hard macros*.
- **Firm macros** are delivered as a gate netlist. They have been structurally optimized for performance, and use a specific semiconductor cell library. They are more flexible and portable than hard macros, yet more predictive on performance and area than soft macros. Protection risk of firm macro is similar to that of the soft macro.
- **Soft macros** are delivered on synthesizable RTL, so they are more flexible than firm and hard macros and are not specific to a manufacturing process. Soft macros have the disadvantage of being somewhat unpredictable in terms of performance, timing, area, or power.

Some considerations have to be made before implementing these kinds of macros. As the hard macros are not flexible in terms of shape and pin location, the floorplanning of the SoC have to take into account these parameters to avoid placement and routing congestion.

1.5.2 Front-End and Back-End Flow

The Front-End and Back-End flow differentiates the main chip design steps. The Front-End is the chip synthesis and verification while the Back-End is the chip layout for fabrication process.

1.5.2.1 Font-End flow

On the Front-End, the VHDL or Verilog RTL (Register Transfer Level) design is synthesized using a standard cell library. The synthesis is the implementation of the circuit with standard cells (gates) while the RTL is the functional description of the

circuit. The synthesis is performed using a timing constraints file (sdc), which defines the timing operating conditions of the circuit.

Once the circuit is synthesized, a gate-level netlist is obtained. The timing constraints are finally verified using a timing check tool.

1.5.2.2 Back-End flow

The synthesized gate-level netlist and the timing constraints are the input files for the back-end. A simplified Back-End flow is depicted in Figure 1.20

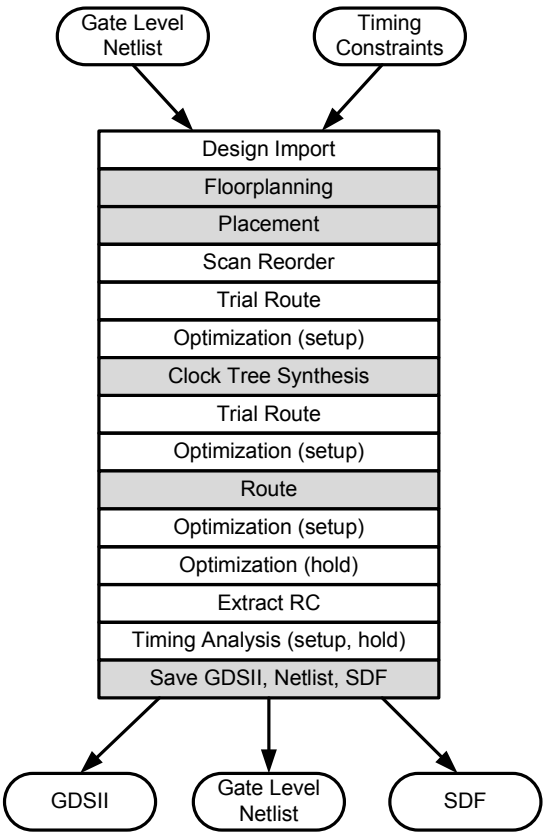


Figure 1.20 Simplified Back-End flow

- **Design Import**: It is the initial step and it is used to setup the environment with the correct technology libraries, load the gate-level netlist, load the timing constraints file (sdc), and configure the tool with the necessary information (buffers/inverters to be used, cells to be do not used, …)
- **Floorplanning**: It is used to define the geometric limitations of the circuit (width/height), to place the input/output/power pins of the circuit, to define the power lines and stripes of the circuit, to define the location and orientation of the hard macros, to define the blocking regions of the

circuit, and to define the regions where the tool should place some modules of the circuit.

- **Placement**: On this step, the tool places all the unplaced cells of the circuit and tries to optimize the placement for the timing constraints while respecting a targeted maximum density. The target density limits the placement density to avoid the wiring congestion of the circuit. Moreover, the tool tries to respect the regions defined on the floorplanning. Hence, all the cells of a module defined with a region are tried to be placed inside its region.
- **Scan Reorder**: The scan chains are reordered to simplify the wiring length and so reduce the wiring congestion.
- **Trial Route**: The circuit is routed with a simplified router to perform a first approximation of the routing complexity.
- **Optimization**: The circuit is optimized on setup/hold time. Therefore, some cells are moved, others are resized, and long wires are buffered.
- **Clock Tree Synthesis**: The clock signal distribution network (clock tree) is synthesized using a configuration script that defines the maximum insertion delay, the maximum skew, and other configuration parameters.
- **Route**: The circuit is routed respecting the DRC rules and minimizing the signal integrity issues.
- **Extract RC**: The resistance and capacitance of the circuit are extracted.
- **Timing Analysis**: The timing analysis is analyzed using the RC extracted data and the timing constraints. The analysis can be performed for the setup or hold time.
- **Save GDSII, Netlist, SDF**: The output files are generated. The GDSII is the layout database for the mask generation. The gate-level netlist of the implemented circuit after Back-End optimization is saved. The SDF file is the timing data used for back annotation simulations.

1.5.3 Clock Tree Distribution and Balance

The *clock tree* is the clock distribution network for a synchronous domain. In a SoC, thousands of synchronous flip-flops are clocked by the same clock signal. Hence, the clock signal has to be distributed and balanced to guarantee a maximum tolerable skew between any pair of flip-flops. The clock tree network is a collection of clock buffers and inverters interconnected in a tree manner. Moreover, all the elements on the clock tree network have to be properly balanced in terms of fan-in

and fan-out to respect the same rising and falling time. The input clock signal arrives to the root of the clock tree while the flip-flops are connected on the leaves.

The clock tree network can be characterized in terms of insertion delay, maximum skew, and power consumption. The insertion delay is the time it takes an event to propagate from the root to the leaves of the clock tree. It depends principally on the number of intermediate buffers/inverters between the root and the leaves and the area covered by the tree. The maximum skew is the maximum difference on time between any pairs of leaves of the clock tree. The lower the skew, the higher complexity of the clock tree and the higher the power consumed. Mesochronous clock tree distributions are well suited for low power consumption and low area. On the other hand, a fully synchronous clock tree networks can consume from 15% to over 45% of the total system power [Qing00] [Chen99] [Chattop05].

1.5.4 Long Link Communications

The reduction of the features sizes enables the design of more complex circuits while preserving the same total chip area. In deep submicron technology the interconnect delays become a major issue. The delays of wires that span the chip will extend longer than the clock period. This phenomenon is the consequence of three phenomena:

- **Resistance and capacitance**: The propagation delay due to the resistance*capacitance (RC) scales up on each technology node. The insertion of repeaters or the current-sensing signaling can be used to minimize these issues.
- **Inductance**: The self and mutual inductance of the wires was neglected on the propagation delay model of the wires. From now on, this additional parameter has to be included on the propagation delay model in order to model the real propagation delay, especially for high clock frequencies.
- **Speed of electromagnetic wave**: Assuming an operation clock frequency of 10GHz on CMOS 50nm [Benini02] and a relative permittivity ε_r of 2 to 3 on the same technology node [ITRS2005], the propagation speed of an electromagnetic wave is $v = (0.3/\sqrt{\varepsilon_r})$. Thus, it is only possible to travel 17 to 21 mm of circuit with one clock period.

Two signaling techniques, voltage-sensing using repeaters and current-sensing, have been used in [Burel05] to characterize the wire delay of 90nm Intel process. Figure 1.21 shows the wire delay in function of the wire length. The width of the wires is two times the minimum width of the process. The current-sensing has better

propagation delay than the voltage-sensing. However, the power consumption of the current-sensing is higher than the voltage-sensing because the link consumes power even when the wires do not toggle.

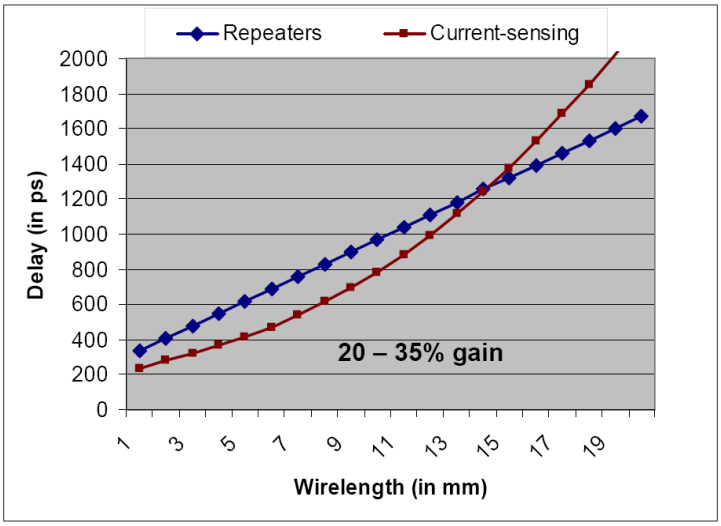


Figure 1.21 Wire delay vs. wire length on CMOS 90nm [Burel05]

1.5.5 Power Aware Design

According to the ITRS [ITRS05] prediction, the power consumed by the interconnect will be 50 times larger than the power consumed by the logic gates [Dally02].

The power consumed by an electronic circuit can be split in two parts:

- **Leakage** is undesired power consumption due to a quantum phenomenon where mobile charges tunnel through an insulating region. The leakage consumption has not any relationship with the circuit activity. It depends on the technology process, the transistor design, the operating voltage, and the temperature.
- **Dynamic** is the power consumption due to the activity of the circuit. It can be divided into three types:
    - **Switching** is power consumption due to change and discharge of the load capacitance.
    - **Internal** is the power consumption dissipated inside the cell for its operation.
    - **Glitch** is the power consumption due to glitch transitions.

The leakage power is somehow related with the cell performance. The speed improvement of a CMOS cell is directly related with the leakage power consumed by the cell. Standard cell vendors generally propose LP or GP standard cells. The LP is

the *low power* library while the GP is the *general purpose* library. However, the introduction of new materials as the High-k dielectric can reduce the leakage power while preserving similar performances.

The clock tree distribution is responsible of 15% to 45% of the total power consumption of the circuit. All the buffers and inverters in the clock distribution network switches when the clock signal toggles. Therefore, clock-gating techniques are suited to cut down the power consumption of the clock tree, and cut down the internal flip-flop consumption. Moreover, the clock-gating cells should be placed as near as possible of the clock root pin to maximize the power saving.

Another source of power consumption is the complexity of the clock tree. The higher the complexity in number of buffer and levels of the clock tree, the higher the power consumed. Consequently, the GALS paradigm is well suited to cut down with the clock-tree power consumption.

A further source of power consumption is the unnecessary switching of wires, specially long and buffered wires. These wires should switch only when useful data is required, otherwise, all the buffering elements of the wire will switch and consume power.

Chapter 2

State of the Art

In this chapter, the state of the art on the Network-on-Chip architectures is analyzed. The range of the analysis is limited to the topics addressed by this thesis: guaranteed service (GS), synchronization issues, and physical implementation. Therefore, architectures that have neither guaranteed service nor been physically implemented are excluded from this state of the art. Moreover, synchronization architectures not implemented on any NoC, are analyzed in Appendix A. The selected NoC architectures are the following:

- **SPIN**: Does not support GS traffic. A 32-port SPIN NoC has been physically implemented.
- **DSPIN**(prior to this thesis): Designed by A. Greiner before the beginning of this thesis, it was the starting point for the thesis. The original architecture is detailed in order to identify the thesis contributions.
- ***Æthereal***: Architecture supporting GS over Time Division Multiplexing.
- **Nostrum**: Architecture supporting GS by looping containers
- **ANOC**: Asynchronous architecture supporting GS using virtual channels. Physically implemented in the FAUST chip.
- **QNoC**: 4-channel router with quality of service NoC.
- **Mango**: Asynchronous NoC supporting GS traffic over virtual channels.
- **Tera-scale**: Multi-processor chip architecture containing a NoC.

2.1 SPIN

The SPIN micro-network architecture was the first published [Guer00] attempt to solve the bandwidth bottleneck, when interconnecting a large number of IP cores in multi-processors SoC. SPIN stands for Scalable Programmable Integrated Network. It was developed by the Université Pierre et Marie Curie.

2.1.1 Architecture

Its architecture is composed of routers (RSPIN) and wrappers (VCI/SPIN and SPIN/VCI). VCI, which stands for Virtual Component Interface, is a SoC interface standard from VSI Alliance [VCI00]. SPIN network uses the fat-tree topology because it is, theoretically, the most cost-efficient topology for VLSI realizations [Leiser85] as shown in Figure 2.1. The routers are packet-based with a flit size of 36 bits. Adaptive routing algorithm and out-of-order delivery can be used to maximize the network bandwidth. Otherwise, deterministic and in-order delivery is used to avoid the reordering buffers on the output ports.

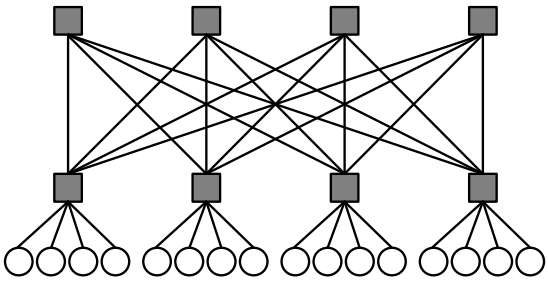


Figure 2.1 SPIN topology

Figure 2.2 shows the SPIN router. It is composed of 8 queues, 2 special queues, and a 10x10 partial crossbar. Special queues (Shared output buffers) are used when a packet cannot be routed due to output port congestion. In this case, the packet is temporarily stored into these queues to allow the others packets to be routed.

Credit based mechanism is used on the wrappers to minimize the network congestion. Moreover, the wrappers have reorder buffers to rearrange the received packets. These buffers have to be properly dimensioned to minimize the circuit area.

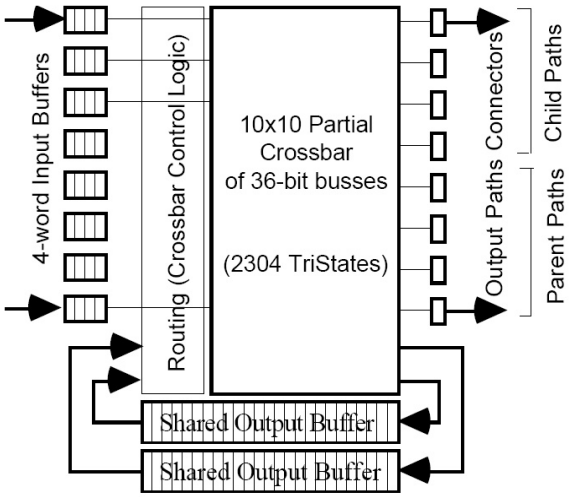


Figure 2.2 SPIN router [Guerr00]

2.1.2 Implementation

A 32-port implementation of the SPIN NoC was done using CMOS STMicroelectronics 0.13 μm technology [Andria03][Andria06].

The implementation of the 32-port SPIN NoC was build using the ALLIANCE [ALLIAN] symbolic layout approach. ALLIANCE is a free suit of CAD tools designed by the University Pierre et Marie Curie.

The design of the 32-port SPIN NoC was build using a hard macro approach. Each router was build and then assembled into a global hard macro. The SPIN router, containing 8 FIFOs, was area-optimized using a data path tool. The highly regular data path of the SPIN routers was implemented using the GENLIB tool (ALLIANCE data path tool) while the control logic was implemented using the Silicon Ensemble automatic place and route tool. Figure 2.3 shows the SPIN router layout. Each SPIN router has an area of 0.24 mm².

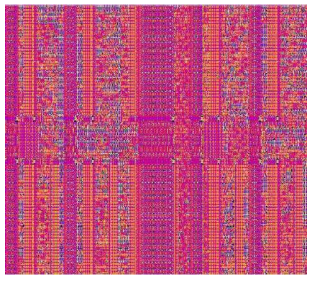


Figure 2.3 SPIN router layout [Andria03]

The assembling of 16 SPIN routers in a fat-tree manner composes the 32-port SPIN NoC. Figure 2.4 shows the 32-port SPIN NoC. Its area is 4.6 mm². The routers are interconnected through metal layers 4 to 6.

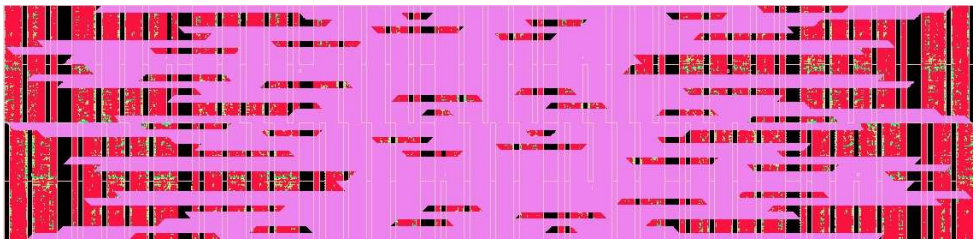


Figure 2.4 32-port SPIN NoC layout [Andria03]

The 32-port SPIN NoC was implemented into the test chip showed in Figure 2.5. The chip was fabricated in STMicroelectronics on CMOS 0.13 μm. It contains traffic generators and analyzers to compute the SPIN NoC performance. This implementation was crucial to reveal the architecture and implementation limitations. The experience gained in this implementation helped to define the DSPIN NoC architecture.

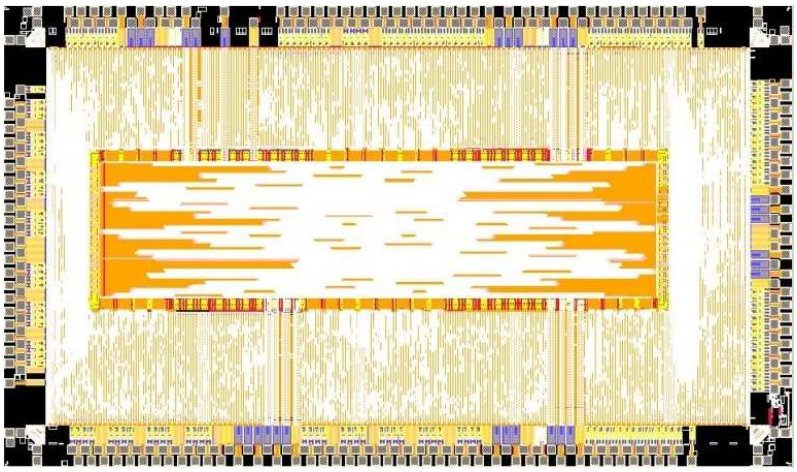


Figure 2.5 SPIN32 test chip layout [Andria06]

2.1.3 Analysis

The SPIN NoC does not have any support for Guaranteed Service traffic. Moreover, the physical implementation of the SPIN network showed several weaknesses and limitations that have been corrected in the DSPIN architecture.

The 32-port SPIN NoC physical implementation was limited by many factors:

- **Flexibility**: The design of the NoC as a hard macro limited the flexibility of the SoC. The centric NoC macrocell conditioned the design and placement of the SoC modules.
- **Timing closure**: Due to the big surface of the SPIN macrocell, the SPIN test chip had many timing closure limitations. The clock trees were

complicated to balance between the macrocell pins and the test chip modules.

- **Tools limitations**: The NoC was built from a unique hard macro cell, difficult to implement in an industrial flow because of its big size. CAD tools have limitations in terms of area of the hard macro and number of input/output pins. The 32-port SPIN macrocell had a 4.6 mm^2 area and more than 2000 input/output pins.
- **Portability**: The portability of the NoC architecture requires redesigning the SPIN macrocell for each technology node as it is a hard macro cell.
- **Scalability**: The fat-tree topology increases linearly with the number of input/output ports; however, it increases in a stairs manner. Example, a 32-port SPIN NoC requires 16 SPIN routers while à 16-port requires 16 routers.

2.2 DSPIN

A first version of DSPIN architecture was designed by A. Greiner before the beginning of this thesis. However, this architecture was never published before the contributions of this thesis. We explain the origin of DSPIN in this chapter rather than on the next chapter where the thesis contributions are explained.

2.2.1 Architecture

The DSPIN architecture is the evolution of the SPIN architecture and it was designed to cope with the GALS paradigm. The main characteristics of the architecture can be summarized with:

- **Topology**: 2D mesh
- **Routing algorithm**: Deterministic X-First routing algorithm
- **Switching algorithm**: Wormhole
- **No deadlock**: Two independent sub-networks are used, one for the request packets and one for the response packets.
- **Clustered multiprocessor architecture**: The architecture is suited to multiprocessor architectures organized in clusters when each cluster can contain one or many processors.
- **Distributed architecture**: Each router has 5 modules placed on the sides (north, south, east, west, local) of the subsystem

- **GALS compatible architecture**: Each subsystem has its own clock frequency. The communications between routers are carried out by bi-asynchronous FIFOs.
- **Synthesizable**: DSPIN is synthesizable with standard cells. Neither custom cells nor asynchronous cells are used.

Figure 2.6 shows the DSPIN cluster architecture and topology. A cluster is the building block of the DSPIN architecture. Each cluster contains two DSPIN routers, one *network interface controller* (NIC), one *local interconnect*, and some computing units (IP). In order to avoid deadlocks in request/responses traffic, DSPIN contains two fully separated sub-networks for requests and responses packets. Therefore, each cluster contains two routers, one for the requests and one for the responses packets. The NIC behaves as a bridge between the IPs and the network while the *local interconnect* router the traffic between IPs of the same cluster. Moreover, any communications between IPs of different clusters have to pass through the DSPIN routers.

The topology of the network is organized as a two-dimension mesh of clusters as shown in Figure 2.6b. Each cluster is connected to the north, south, east and west neighbors by means of point-to-point links. The communication between IPs in different clusters is done by traveling through as many routers as necessary (more precisely N+1 routers, if N is the Manhattan distance between the communicating clusters).

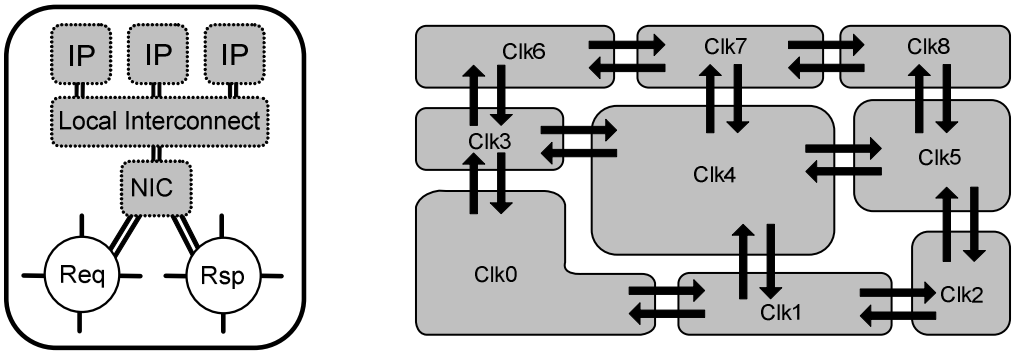


Figure 2.6 DSPIN cluster architecture and topology

The physical links between routers are implemented with FIFOs (black arrows in Figure 2.6b). The mesh topology simplifies the routing algorithm, and strongly minimizes the silicon area of the switching hardware. There is no constraint on the

size or shape of clusters except that the mesh topology has to be respected, to guarantee the routing path between all the clusters.

Each cluster has its own clock signal, which can be different in terms of frequency and phase from the neighbor clock signal. Moreover, the IPs and DSPIN routers on the cluster share the same clock signal. Therefore, the FIFOs interconnecting routers are asynchronous while the FIFOs interconnecting router-to-NIC are synchronous. Under these circumstances, the GALS approach can be used because each cluster is synchronous but asynchronous compared to its neighbor. Thereby, an independent clock-tree is synthesized per cluster.

The DSPIN router is not a centralized macrocell: it is split in 5 separated modules (North, East, South, West & Local), that are physically distributed on the clusters borders (Figure 2.7). This feature, combined with the mesh topology allows us to classify the network wires in two classes:

- **Inter-cluster wires** connecting modules of adjacent clusters. Example: the East module of cluster (Y,X) is connected to module West of cluster (Y,X+1). As those components can be made very close from each other, inter-cluster wires are short wires.
- **Intra-cluster wires** connecting modules of the same cluster. Example: West module connects to North, South, East and Local modules in a tree manner. Those wires are the long wires, but the wire length is bounded by the physical area of a given synchronous domain, the cluster.

These properties allow synthesizing, placing and routing each cluster as an independent module. Moreover, the design relies on standard synchronous design flow without custom cells.

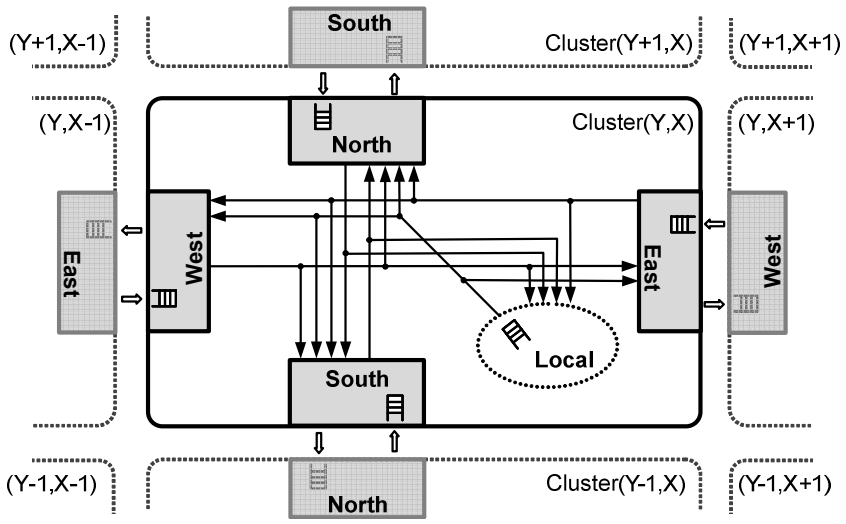


Figure 2.7 DSPIN router architecture

DSPIN was originally designed to interconnect IPs using the VCI/OCP [VCI00] protocol. The network interface controller converted the VCI request/response packets format to DSPIN packet format. For this reason, the DSPIN packet format was designed specially for the VCI protocol. The flit size is 36 bits with a payload of 32 bits. The flit control bits are Begin of Packet (BOP), End of Packet (EOP), Parity error (PAR), and Packet error (ERR) as shown in Figure 2.8. *BOP* is set on the head flit while *EOP* is set on the tail flit to identify the beginning and ending of a packet. The minimal packet length is one flit. The *Parity error* bit identifies an error at the network level; for example, a crosstalk error bits, or a soft error on the DSPIN router. The *Packet error* identifies an error on the VCI protocol without any relationship with the network; for example, a request to an address not mapped on the architecture, or a write operation to a read-only register.

In order to router the packets on the network, the head flit includes the destination cluster address defined in absolute coordinates Y and X, encoded on 4 bits each one, allowing a maximal 16 * 16 = 256 clusters topology.

The switching hardware in each module (North, East, South, West & Local), is composed of one multiplexer controlled by one state machine (Figure 2.8b). Due to the X-first routing algorithm, the multiplexers for the East and West modules are reduced to simple (2 inputs to 1 output) multiplexers, while for the North, South, and Local they are longer (4 inputs to 1 output). This comes from the fact that the packets coming from North port can be routed to neither East nor West port.

When the router receives the first flit of a packet, the destination field is analyzed and the flit is forwarded to the corresponding output port. As DSPIN uses wormhole routing, the rest of the packet is also forwarded to the same port until the tail flit.

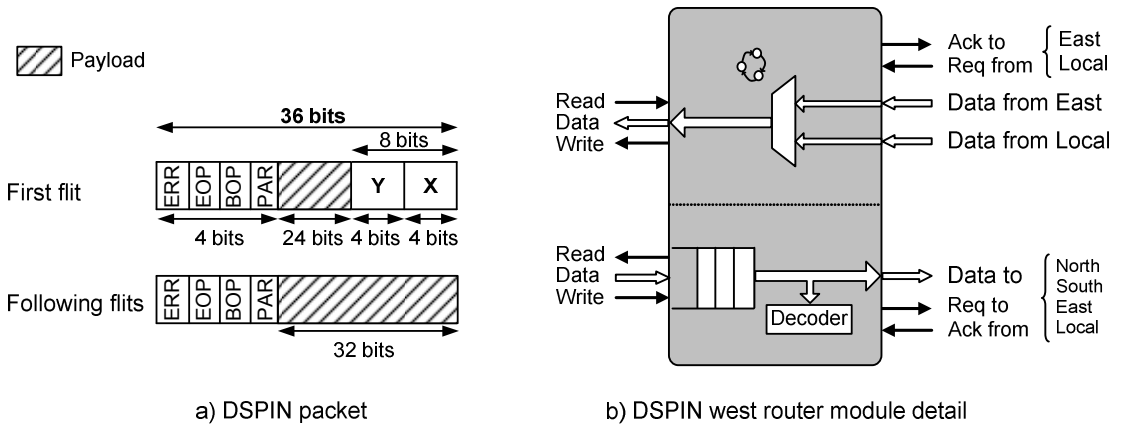


Figure 2.8 DSPIN packet and west router module detail

2.2.2 Analysis

This initial version of DSPIN can be used for best effort traffic. However, DSPIN does not support guaranteed service traffic because it does not have any resource reservation for guaranteed service traffic. The introduction of the guaranteed service traffic is detailed in Chapter 3.

Regardless of the unavailability of the guaranteed service traffic, the architecture is simple and well suited to the GALS approach.

The packet format does not follow the ISO-OSI reference model, as it uses control bits from the VCI protocol. These bits should be placed on the payload of the DSPIN packet rather than on the DSPIN packet format. The DSPIN packet format is modified on Chapter 3 in order to respect the ISO-OSI reference model.

2.3 Æthereal

Æthereal NoC is an NoC developed by Philips offering both Guaranteed Service and Best Effort traffic [Goos05][Radu05].

2.3.1 Architecture

The router uses a contention-free routing mechanism to send independent traffic on the same physical links. Therefore, a time-division multiplexing mechanism is used over the physical links to send independent traffics. On each router, a reconfigurable table is used to switch the GS traffic to the correct output while avoiding contention on the link. Every reconfigurable table T has S time slots (rows) and N output ports (columns). There is a logical notion of synchronicity, since all routers in the network are assumed to be in the same fixed-duration slot. Figure 2.9 shows an example of contention-free routing. Packet A and B are routed without contention between router R1 and R2 because they use different timing slots. In the same way, packets A and C do not have contention in between routers R2 and R3. The reconfiguration of these tables are carried out by special packets sent over the BE network.

The contention-free mechanism requires a synchronicity of all the network tables as well as no stalled packets over the network. In order to guarantee this last condition, Æthereal uses end-to-end credit-based mechanism.

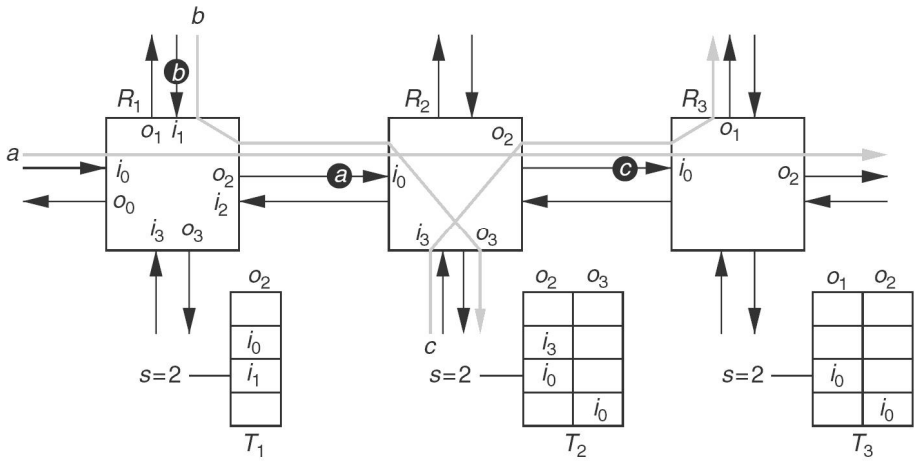


Figure 2.9 Æthereal contention-free routing [Goos05]

Æthereal NoC can be designed for a distributed or a centralized programming model. On the distributed programming model, special BE packets are sent to the router to request the allocation of a GS traffic. The router is able to decide if the GS packet can be routed or not. On the centralized programming model, the slot tables are located on the Network Interface and no longer on the router. Moreover, a centralized slot allocator decide which GS packets can be served by sending BE packets to the Network Interfaces in order to configure the timing slots. Consequently, the complexity of the router is moved to the NI.

The BE packets are routed using round-robin arbitration, wormhole routing, input-queuing, and source routing. Figure 2.10 shows the Æthereal packet format. The flit size is three words of 32-bit and 2 control bits. The first word contains the routing information in 22 bits, the piggybacking credits on 5 bits, and the destination queue in 5 bits. Æthereal NoC uses credit-based flow control to minimize the network contention and avoid the deadlock situations. To minimize the credit traffic, the returned credits are sent using the response packets (piggybacking) over 5 bits. Thus, the maximum number of credits that can be sent at a time is $2^5 = 32$.

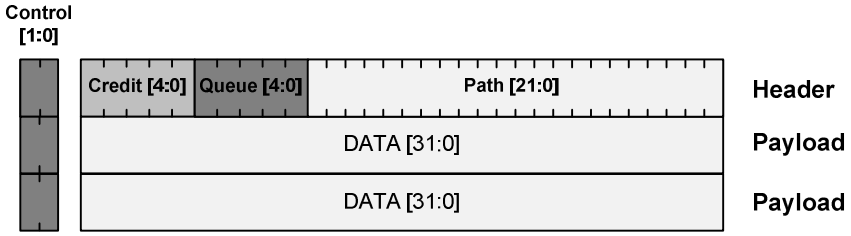


Figure 2.10 Æthereal packet format

Both traffics are multiplexed over the same resources to maximize the bandwidth utilization. The unused bandwidth (unreserved or reserved but not used) is employed by the BE traffic. The BE flits, having lower priority, can use a link only when there is no GS flit on the link.

2.3.2 Implementation

The design and implementation of the Æthereal NoC uses an automatic design flow (Figure 2.11) [Goos05b]. The flow is used to design the network topology, router arity, and table size. Moreover, the designed architecture can be fully simulated in SystemC and RTL VHDL. Thus, for a specific application it is possible to compute the packet latency, the interlocking issues, and the power consumption estimation. At the end of the flow, a RTL VHDL code is obtained for synthesis.

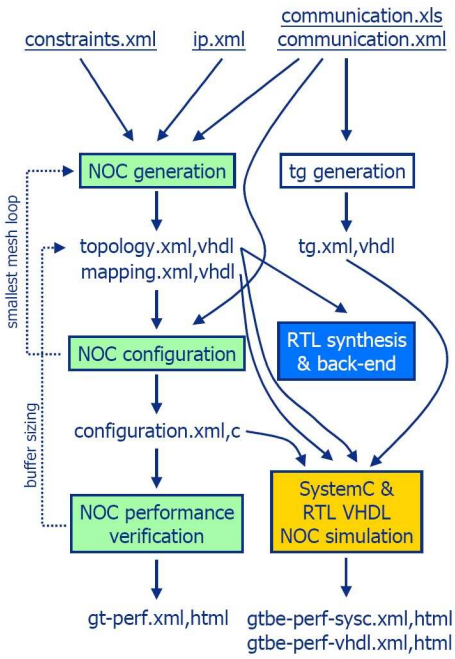


Figure 2.11 Æthereal NoC design flow [Goos05b]

In [Goos05], a 6-port Æthereal NoC implementation is detailed on CMOS 130nm technology. The router has 6 ports; however, just 4 of theme are used for inter-router connections. A distributed and a centralized programming architecture are described on the paper. Both of them are designed as a hard macro with dedicated hardware FIFOs for the BE and GS queues. Moreover, a dedicated hardware slot table is used on the distributed programming architecture for the congestion-free routing algorithm. These dedicated hardware devices are designed to minimize the router area. Figure 2.12 shows the distributed and centralized router architecture

implementation. The hardware dedicated FIFOs and slot table are depicted on the squares. Moreover, the distributed programming router architecture contains the *Reconfiguration unit* (on its left side). It is used to dynamically allocate and desallocate the GS traffic.

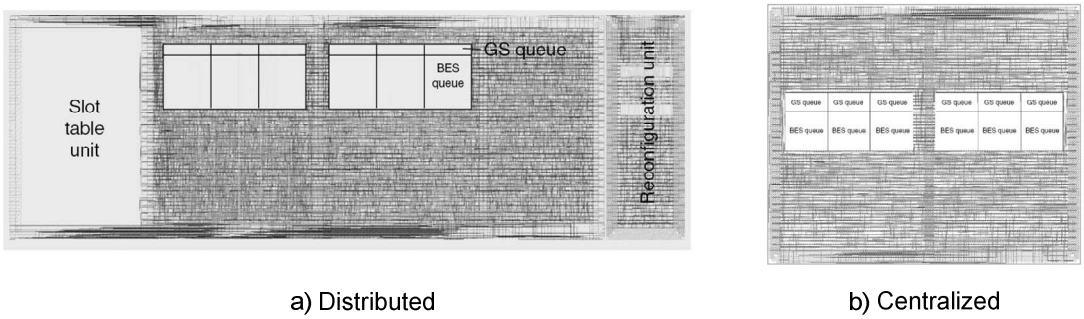


Figure 2.12 Implementation of GS-BE Æthereal distributed and centralized programming router architecture [Goos05]

Table 2.1 shows the area and frequency of the distributed and centralized programming router architecture [Goos05]. The area of the centralized architecture is smaller than the distributed version because neither *Slot table* nor *Reconfiguration unit* are used.

Table 2.1 Distributed and centralized comparison

	Area	Frequency
Distributed architecture	0.24 mm²	500 MHz
Centralized architecture	0.13 mm²	500 MHz

The Network Interfaces are designed using the Æthereal NoC flow. Its architecture contains custom-made hardware FIFOs to be area efficient. Moreover, clock boundaries are defined on each NI port to run at different clock frequencies while the NI-kernel work at the router frequency (500MHz on CMOS 0.13 μm). Its area is conditioned with the number of FIFOs, the FIFO depth, and the NI available services (multicast and narrowcast).

The Æthereal NoC design flow was used to implement a MPEG codec SoC with 16 IPs [Goos05b]. Four case studies were analyzed: fully automatic, naive mapping, simulation, and optimized. The naive mapping uses one IP per NI while the simulation implementation, oversized the FIFOs for the worst-case condition. Table 2.2 summarizes the silicon area of the NI and router for these case studies.

Table 2.2 Comparison of Æthereal router for MPEG SoC

	Mesh	Table slots	NI area (mm^2)	Router area (mm^2)	Total area (mm^2)
Automatic	2x3	128	1.83	0.51	2.35
Naive	3x6	128	2.17	2.32	4.49
Simulation	2x3	128	4.61	0.51	5.13
Optimized	3x1	8	1.51	0.35	1.86

2.3.3 Analysis

The contention-free routing mechanism is a coherent approach to route the GS without contention over the network. However, this mechanism requires two conditions:

- **Synchronicity**: All the routing tables have to be synchronized. Otherwise, some flits will be stalled over the network thus blocking the other packets. In this situation, the routing mechanism is not able to guarantee a congestion-free situation. The design of a fully synchronous network is not a scalable solution due to the limitations on the clock tree distribution. Therefore, the authors propose waterfall clock distributions [Goos05] and a Synchronous Latency Insensitive Designs (SLID) [Ru06]. Under these circumstances, each router synchronizes every slot all its neighbors. Thus, all routers always remain in the same slot and the NoC run as fast as its slowest router.
- **Contention-free consumer**: The contention-free routing mechanism requires a contention free consumer. Any consumer in the network should be able to consume all packets addressed to him. The authors propose to use an end-to-end credit-based flow control mechanism in order to guarantee this condition. Otherwise, the contention-free mechanism does not work properly. The end-to-end flow control requires deep FIFOs on the NI in order to deliver 100% throughput. Example, if the producer-consumer path takes 20 clock cycles, the FIFO depth on the NI should have 2 * 20 = 40 words to guarantee 100% throughput.

Taking into consideration these two aspects, the Æthereal contention-free routing mechanism is an efficient method to deliver guaranteed service traffic. The storage elements for BE and GS traffics are independents and different GS traffics do not share the same timing slots. Therefore, no contention situation occurs on the GS network. Moreover, the GS throughput is guaranteed. The more the timing slots

assigned to a GS communication, the higher the throughput. In terms of jitter latency, it is very low because it is constrained by forced synchronicity of the contention-free routing mechanism.

The allocation of a GS communication channel is established like a circuit. The full end-to-end path has to be established before any GS communication packet can use it. Moreover, the configuration of routers/NI uses BE packets which can be delayed due to network congestion. Therefore, the allocation and desallocation of the GS channels can be highly impacted by the network congestion. This network characteristic can influence on the performance of an application if it often allocates and desallocates the GS traffics.

In terms of GS traffic types, the congestion-free routing algorithm is extremely efficient for regular and deterministic traffic. This steams from the synchronicity of the whole architecture. However, when burst or non-regular GS traffic is used, the number of allocated slots for these traffics has to be oversized to overcome the worst-case condition. Thus, the allocated bandwidth for other GS traffics is reduced.

The Æthereal NoC design flow is explained in [Goos05b]. However, no detailed analysis of its physical implementation is showed. In [Bartels06] and [Steenh06], two design implementations are analyzed; however, their architectures are analyzed at VHDL RTL and SystemC.

Æthereal NoC routers uses hardware optimized FIFOs and tables to be an area-optimized design. However, these hardware devices become a constraint in the physical implementation, as they have to be placed and routed as a hard-macro (Figure 2.12).

The Æthereal routers are designed as independent hard-macros and later placed and routed with the rest of the design [Goos05]. The design of a complex SoC, requires optimizing each Æthereal router, synthesizing router-by-router, placing and routing each Æthereal router, and finally assembling the SoC. This comes from the fact that the NoC design flow optimizes each Æthereal router (ports, FIFO depth, and routing slots) of the SoC. These optimizations are very time consuming in terms of Back-End implementation cost because each Æthereal router would require different hardware-dedicated devices. Moreover, each router has to be synthesized, floorplanned, placed and routed as an independent unit. Example, the design, implementation and verification of a 2*2 Æthereal NoC required 12 person months effort [Steenh06].

The NI detailed in [Radtu05] has independent clock domains for the IPs and the router. Therefore, it is possible to use independent clock frequencies on each IP.

However, the Æthereal routers require a clock synchronicity between all the routers of the network [Goos05]. The routers synchronize with their neighbor routers by means of the flow control signals. Therefore, the routers run as fast as the slowest router.

A waterfall clock distribution and synchronous latency-insensitive design is supposed to be used on the synchronization of the routers [Goos05] [Ru06]. However, these techniques can influence the efficiency of the congestion-free routing because the synchronicity of routers can be influenced by the process variation, temperature, voltage, and operation frequency. Example, when a router to router interface can not exchange data without metastability (out of the metastability window), an additional clock cycle have to be waited to guarantee the data correctness. Thus, the congestion-free routing algorithm has an additional penalty of one clock cycle and all the remaining communications have to be recalculated.

The area of the router plus NI is too high for the target applications that we are considering. Its total area is about 2 mm^2 on CMOS 130 nm. Assuming an NoC implementation cost per cluster of 15% of the total area, the area of the cluster would be 13mm^2 while we are targeting clusters of 5 mm^2.

2.4 Nostrum

Nostrum is an NoC developed by the LECS (Laboratory of Electronics and Computer Science) at the Royal Institute of Technology in Sweden [NOSTR].

2.4.1 Architecture

The Nostrum NoC architecture follows a regular mesh topology containing switches and network interfaces. Two traffic classes are available, Best Effort (BE) and Guaranteed Bandwidth (GB). In the BE implementation, the packet transmission is handled by datagrams. The switching decisions are made locally in the switches on a dynamic/non-deterministic manner by means of the deflection routing algorithm. Its benefits are robustness against network link congestion and link failure. However, the BE packets may arrive in another order that they were sent; thus, the NI handles the ordering of packets and de-segmentation of messages. The BE packet size is one flit.

The defection routing algorithm guarantees that no packet is stalled in the router, thus no intermediate buffer is required in the network. All packets in the switch are forwarded to an output port; even it is not the requested one. This phenomena

requires that the entire network is synchronized (all switches have the same clock frequency and switch at the same clock cycle).

The GB traffic is handled using containers [Millbe04]. A container is a network packet that follows a predefined looping path as shown in Figure 2.13. They can transport the information of GB traffic; but if they do not transport any information, they continue to follow the predefined looping path. Thus, they contain an *empty* flag to identify if they transport or not data.

Figure 2.13 illustrates an example of a lopping container when a GB source transfers packets to its GB destination. When the empty container arrives to the switch 1 (the GB source), the GB source load the container with the GB traffic and sent it to the east switch (blue line). The container and its load is routed though the network following its predefined looping path. When it reaches the GB destination, the container is unloaded and it is sent back (red line) empty, possibly, with some new information loaded.

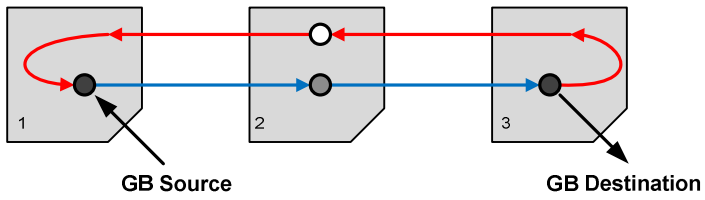


Figure 2.13 Nostrum looping containers

When looping containers are temporally multiplexed, the network is able to sent different GB traffics over the same link. Figure 2.14 shows an example of bandwidth sharing between two independent GB traffics.

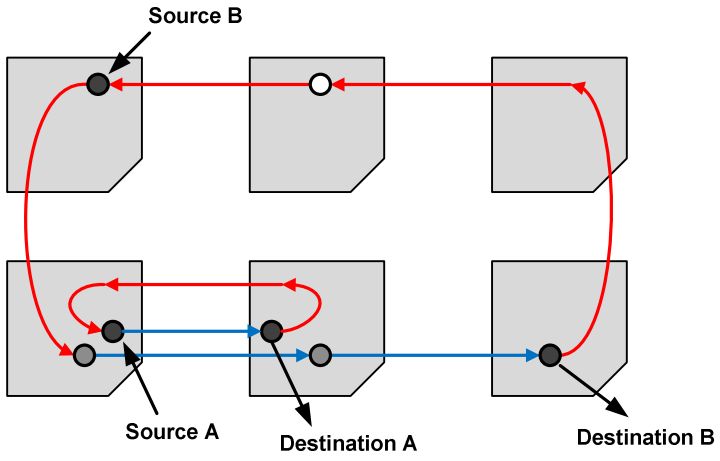


Figure 2.14 Nostrum bandwidth granularity

The containers are launched on the start-up phase of the network when no BE packets are allowed to enter the network. The higher the bandwidth required, the higher the launched containers over the same loop.

2.4.2 Analysis

The looping containers method guarantees the bandwidth of the GB traffic; however, its implementation is not efficient:

- **Missed bandwidth**: The empty containers cannot be used by the BE packets. Thus, the container bandwidth is lost when it is not used. Moreover, the bandwidth of the containers between *destination* to *source* path is always loosed if the *destination* module does not send back any data.
- **Burst packets**: The loping containers are efficient for constant bit rate transmissions. However, under burst operations or variable bit rate, the number of containers has to be dimensioned on the worst-case condition, thus losing bandwidth.
- **Synchronicity**: Routers have to be switched on the same clock cycle to guarantee the timing multiplexing of different GB traffics. Otherwise, it is possible that different GB traffics could not share the same link.
- **Deflection routing**: Nostrum use the same resources for the BE and GS traffics. However, the BE traffic uses the deflection routing algorithm to avoid congestion situations. Therefore the BE packets have to be reordered on destination by reordering buffers which are high area consuming devices.

In terms of throughput, the GB throughput depends on the number of containers following the same predefined path. The higher the number of containers, the higher the allocated throughput.

The allocation of the containers has to be performed at the beginning, where no BE packet is in the network. The allocation of containers for different GB communications has to be synchronized in order to schedule the containers without collision.

2.5 ANOC

ANOC stands for Asynchronous NoC and it is developed by the CEA-Léti. The NoC has been physically implemented in the FAUST chip, a stream-oriented multi-application platform for 4G telecom.

2.5.1 Architecture

ANOC is a packet-based wormhole network-on-chip. Its router has five bidirectional ports to the North, South, East, West, and Local connections (Figure 2.15). The interconnections between routers are bidirectional links using asynchronous send/accept handshake protocol. As the ANOC routers are asynchronous, the entire end-to-end path between the packet producer and the consumer is completely asynchronous. Just the local input and local output ports are synchronized to the subsystem clock frequency. Moreover, a four-phase Quasi Delay Insensitive (QDI) protocol is used on the network guaranteeing no metastability issues inside the router. Just the local input and output ports where synchronization to the local clock frequency is required are susceptible to metastability failure. These interfaces use special FIFOs to minimize the metastability failure.

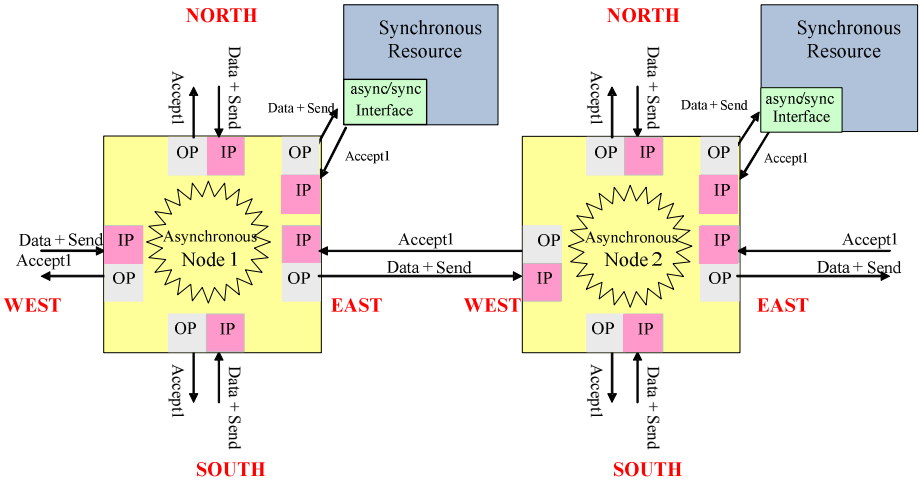


Figure 2.15 ANOC node architecture

The ANOC router does not impose a regular topology for the network. Irregular topologies can be implemented, as ANOC uses a source routing algorithm. This particularity allows a higher flexibility on the routing of the packets over the network. However, it requires complex configuration of the Network Interfaces Controllers and a higher packet overhead to carry the routing information. Packets are subdivided in 34-bit flits. The first flit carries the routing information on 18 bits.

Two bits encode each routing hop as shown in Figure 2.16. Hence, the routing path is limited to nine hops H0 to H8. However, a path extension mechanism is also proposed to extend the routing path [Beigne05]. These routing extension paths have been implemented on the FAUST platform and they are depicted as *EXP* modules on Figure 2.17.

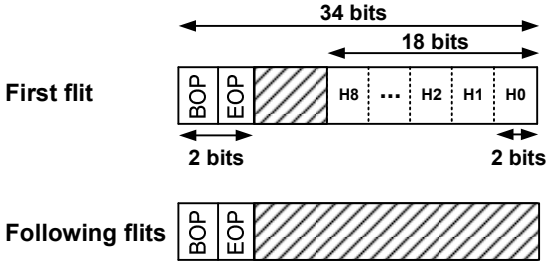


Figure 2.16 ANOC packet

The ANOC architecture provides two virtual channels per physical link (inter router wires). A low latency and high priority channel VC0 and a higher latency and low priority channel VC1. The VC0 channel is intended to be used on real-time applications while the low priority VC1 is used for best effort traffic. VC0 has higher priority than VC1 and can suspend the path of this last one. A VC1 packet can only be suspended by VC0 packets with higher priority. In that case, the suspended packet is stalled and stored in previous nodes.

The allocation policy of the outputs ports is not equal for VC0 and VC1 channels. For the VC1 channel, it uses a "first arrived, first served" (FAFS) allocation policy, while for VC0 channel, it uses static arbitration (N,S,E,W,Res). These allocation policies are simple to design on asynchronous circuits and have faster execution time rather than a round-robin allocation policy.

Each router is composed of 5 *input controllers* and 5 *output controllers*. Each *input controller* is connected to only 4 *output controllers* because back and forth on a same network link is not allowed by the communication protocol. The interconnections between input and output controllers are similar to those of a 5/5 crossbar. The *input controller* can store two flits per virtual channel and its flow control is credit-based. A packet can be sent over the virtual channel, only if the *input controller* has at least one free register.

2.5.2 Implementation

The ANOC was implemented on the FAUST demonstrator platform. FAUST, which stands for Flexible Architecture of Unified Systems for Telecom is a hardware

demonstration platform for the 4MORE mobile terminals. 4MORE [Kaiser04] is an IST program targeting 4G baseband modem chips. The FAUST project was initiated in 2003 for supporting multiple OFDM air interfaces in a single SoC. FAUST architecture is composed by processing units interconnected by a NoC. It also includes an ARM946ES in an AHB subsystem. The communication protocol between the functional units is carried out by message passing through the NoC. Each processing unit contains a programmable Network Interface Controller, which contains input and output FIFOs and regulates the traffic through the network. This regulation is carried out by credits to synchronize the producer to the consumer on a self-synchronized data pipeline manner.

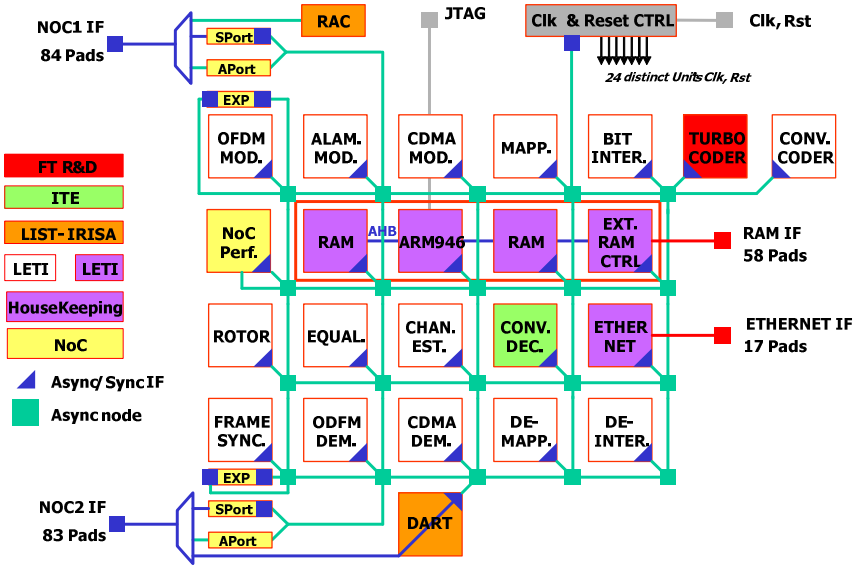


Figure 2.17 FAUST architecture

The FAUST chip is a multi-application platform for 4G telecom. It can support OFDM-based applications such as 802.11a standard, MC-CDMA [Kaiser04][Berens05] and 3GPP-LTE protocols. All these applications share the same set of constraints, including real-time requirements, high throughput and low power consumption for battery-powered devices.

The ANOC design has been implemented in the STMicroelectronics 130nm technology, using standard place-route tools (Encounter™ from Cadence).

For the ANOC router, a hard-macro approach was defined in order to re-use the ANOC router all over the FAUST top floor-plan. This choice allows proper placing of the ANOC router port signal pins (North, East, South, West, Unit). The ANOC router contains robust QDI 4-phase/4-rail asynchronous logic [Beigne05], which is implemented using standard-cells and specific C-elements from the TAL library

[Mauri03]. Once the ANOC router hard macro was available, the standard abstract and *gds* files were generated. For the GALS interfaces implementation, a soft-macro approach was defined.

For top-level, the complete floor-planning was done in order to place all the hard-macros: ANOC routers, SRAM memories, ARM946 core (Figure 2.18). The place & route was done hierarchically with five distinct partitions using Encounter tool. The timing analysis and optimization of the NoC links was possible using a pseudo-synchronous timing model of the ANOC router. For GALS interfaces, timing optimization is more difficult due to mix-timing constraints of these interfaces [Beigne06].

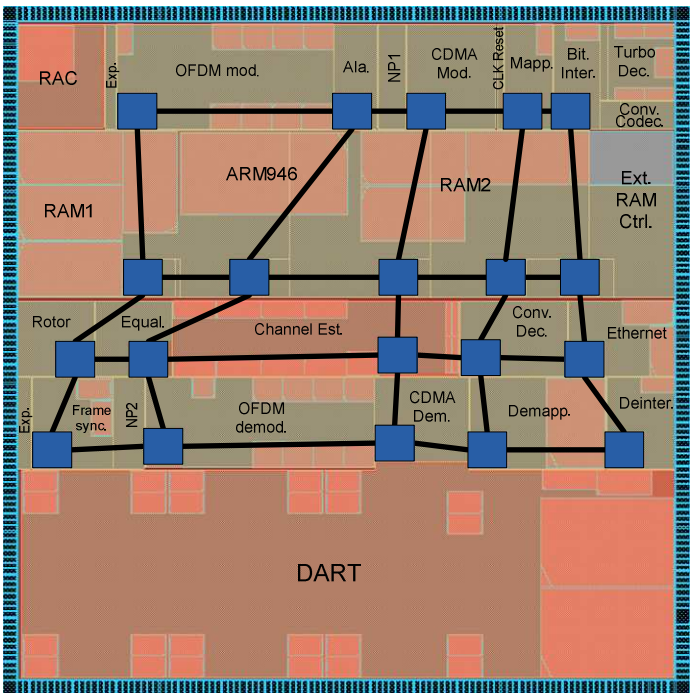


Figure 2.18 FAUST floor-plan with ANOC

Due to the GALS approach on the chip design, the clock-tree of the chip was constituted of 27 independent clock trees: one distinct clock tree per synchronous IP unit. The 27 clock-trees were then generated one-by-one by the tool.

2.5.3 Analysis

The ANOC router uses the virtual channel approach to combine the Best Effort and the Guaranteed Service traffics. Thus, guarantees in terms of latency and bandwidth can be achieved as ANOC uses:

- Independent storage elements for BE and GS traffic classes.

- Credit-based end-to-end flow control to avoid blocking the virtual channels.
- Priority allocation policy for the VC0 channel.

However, the arbitration policy between virtual channels is restrictive because "a low priority packet can be suspended by higher priority packets". It means that the low priority packets can be blocked as long as a high priority packet uses the virtual channel. This condition can incur in a starvation situation where the low priority packet cannot reach its destination because it is always suspended by a high priority packet. Moreover, this condition can become a deadlock situation when a high priority packet cannot be served until a low priority packet has finished to be received, this last one suspended by the high priority one. An example of this phenomenon is depicted in Figure 3.5 and explained in Chapter 3. This limitation is overcome when an end-to-end credit-based flow control Network Interface Controller [Cler05] is used. With this mechanism, a high priority packet has always enough FIFO space to enter into the destination FIFO, thus, preventing a low priority packet to be blocked indefinitely.

The ANOC architecture is fully asynchronous and requires special libraries to be implemented. These libraries are not currently available in industrial flows; thus, limiting the portability of the design.

In terms of testability, the asynchronous circuits are very difficult to test due to the causality of the circuit signals. They require exhaustive test to verify the correctness of the circuit.

In terms of physical implementation, the ANOC router has been physically implemented as a hard-macro. Thus, the flexibility of the circuit floorplan is reduced because the router itself became an additional constraint in the floorplanning of the circuit. Moreover, the inter-router communication uses 4-phase/4-rail QDI. Thus, hard-macro has more than 900 input/output ports to be connected to other routers.

The ANOC router throughput depends on the optimized physical implementation of the circuits. The inter-router links are implemented by wires and buffers; no intermediate pipeline module is implemented. Thus, the long wire delays dramatically penalize the router throughput. This comes from the fact that a 4-phase QDI asynchronous transaction is performed after 4-phase transaction. Consequently, the critical path in-between two ANOC routers cross four times the long wires between routers.

A detailed comparison after physical implementation between the ANOC and DSPIN NoC in terms of router area, latency, throughput, and power consumption is analyzed in Chapter 5.

2.6 QNoC

QNoC stands for Quality of Service Network-on-Chip and it is developed on the Electrical Engineering Department, at the Israel Institute of Technology.

2.6.1 Architecture

The network architecture is based on a grid topology that can be irregular. The routing algorithm is XY and YX, therefore the network traffic is distributed non-uniformly over the mesh links, but each link bandwidth can be adjusted to its expected load. The links bandwidth can be modifying by sizing the number of data wires or modifying the link frequency [Bolotin04].

The router has five ports and uses wormhole routing algorithm. The inter-router communication uses credit-base flow control. These credits are sent using specific wires to the neighbor router. The links use handshake interfaces and can be adapted for asynchronous interfaces.

In order to support different classes of QoS for different kinds of on-chip traffic, QNoC has four types of service levels (SL). A service level is a traffic class with a common QoS. For example, consider the following four different SLs: *Signaling* (urgent short packets that have the highest priority), *Real-Time* (guaranteed bandwidth and latency to streamed audio and video), *Read/Write* (short memory and register accesses), and *Block-Transfer* (long messages such as DMA transfers) [Guz07]. The service level priorities are ranked with Signaling having the highest priority, Real-Time being second, RD/WR third and Block-Transfer ranked last.

Figure 2.19 shows the QNoC router architecture. Each input port is connected to 4 queues (one per service level) through a demultiplexer. A crossbar interconnects the input ports to the output ports. The CRT (Current Routing Table) and CSIP (Currently Serviced Input Port) modules control crossbar allocation. The output ports are composed of four one-flit storage elements (for each SL), credit counters (NBS), and a control module. This last module receives the neighbor routers credits, updates the NBS counters, and controls the allocation of the output port.

The current state of round-robin scheduling is stored in the Currently Serviced Input Port number (CSIP) table for each service level at each output port. This

number is advanced when transmission of a complete packet is finished or if there is nothing to transmit from a particular input port and service level. This scheduling discipline implies that a particular flit gets transmitted on an output port as long as there is buffer space available on the next router and there is no packet with a higher priority pending for that particular output port. Once a higher priority packet appears on one of the input ports, transmission of the current packet is preempted and the higher priority packet gets through. Transmission of the lower priority packet is resumed only after all higher priority packets are serviced [Bolotin04].

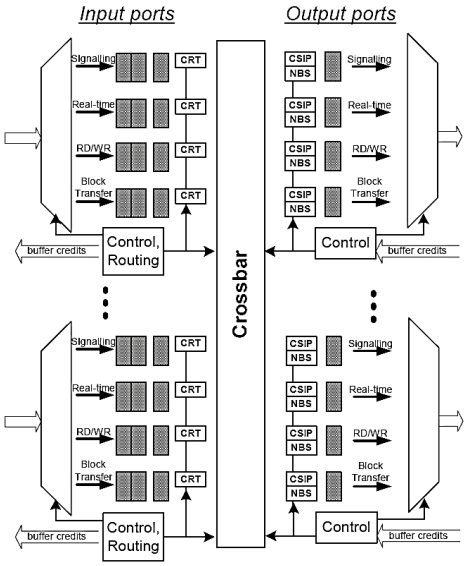


Figure 2.19 QNoC router architecture [Bolotin04]

A flit is transferred from the output router port to its neighbor router input port when the input router port has at least one free place (of the required SL). This mechanism is carried out by the credit-base flow control. The input queues require at least a depth of four flits in order to maximize the throughput. This number is calculated using the cycle type of the router [Bolotin03]:

1. One clock cycle is required for transmitting the flit.
2. One clock cycle is required for latching incoming flit and routing decision in the router
3. One clock cycle is required for the transmission delay of credit-buffer information from the next router.
4. One clock cycle is required for latching the credit-buffer information in the scheduling logic of the output port.

The QNoC router has been implemented in two manners, asynchronous cells and synthesized on synchronous 0.35μm standard cells [Dobkin05]. The asynchronous

implementation introduces naturally the asynchronous communication between routers while the synchronous has to guarantee the correct operation without metastability. Two different architectures are analyzed, one with just one SL and 8-bit flits, and another with four SL and 10-bit flits. Table 2.3 summarizes the results.

Table 2.3 Comparison results of QNoC implementation [Dobkin05]

Parameter	Synchronous Router		Asynchronous Router		Units
	1-SL	4-SL	1-SL	4-SL	
Cell Area	0.210	0.960	0.093	0.470	mm^2
Number of FFs / Latches	195	880	130	620	
Min Latency (Input to Output)	3.3 (1)	3.7 (1)	7.6 / 3.9	13.0 / 9.2	ns (CLKs)
Data Cycle	13.2 (4)	14.8 (4)	18.0 / 11.9	13.3 / 13.3	ns (CLKs)
Max Data Rate	75.8	67.6	55.5 / 84.0	75.2 / 75.2	Mflits/s
Max Clock Frequency	303.0	270.2			MHz

2.6.2 Analysis

The QNoC architecture has four independent channels multiplexed over the inter-router wires. These channels can be used for urgent messages or guaranteed service communications. The multiplexing of these channels follows a virtual channel approach with an independent buffer per channel. The channels scheduling is not static as it depends on the channels priority. The main weakness of this architecture is that a low priority communication can be stalled by a higher priority channel and only resumed after all higher priority packets are serviced. This condition can induce starvation situations of the low priority channels when the higher priority communication does not grant the channel.

The proposed architecture is designed as a macro cell router; no distributed implementation is possible as the internal crossbar is complex. The crossbar switch interconnecting the input ports to the output ports requires five independent crossbars of 4-input 4-output, one crossbar per SL. Therefore, these crossbars can induce wire congestion on the design of the router.

The synthesis of the QNoC architecture on standard cells showed a compact and fast implementation of the router. However, the maximum throughput of the router (Max data rate) is 4 times lower than the maximum clock frequency. Therefore, the architecture is not balanced in term of clock frequency and throughput. Under these circumstances, the power consumption of the clock tree will be higher than the power consumption of the router itself (see Chapter 5 and Appendix C for further

details). An optimized architecture should have a throughput equal to the clock frequency (1 flit per clock cycle). Consequently, the clock frequency can be lowered as much as possible and achieve similar performances as QNoC while the clock tree power consumption is reduced.

2.7 MANGO

MANGO is the NoC developed at the Technical University of Denmark (DUT). MANGO stands for Message-passing Asynchronous Network-on-Chip providing Guaranteed services through OCP interfaces. The architecture supports Best Effort and Guaranteed Service traffic over a virtual channel approach.

2.7.1 Architecture

MANGO is an asynchronous NoC architecture where the routers are the nodes of a 2D mesh. A router has five ports where one is a local port. The router consists of a BE router, a GS router output buffers, and link arbiters (Figure 2.20).

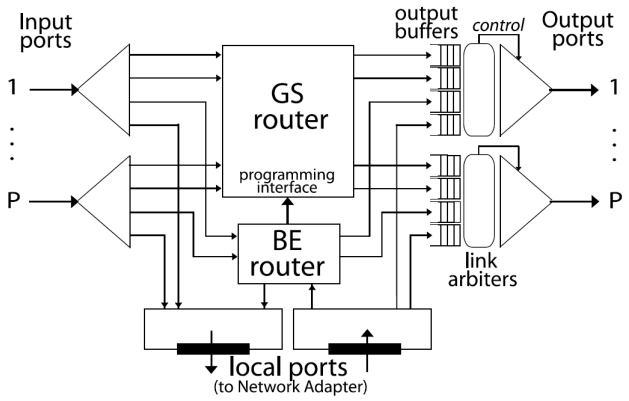


Figure 2.20 MANGO router [Bjerre05a]

The BE router implements a source routing scheme. The first flit contains the routing information. The two MSB bits of the first flit indicates one of the four output ports. When the packet is routed, the packet header (first flit) is rotated two bits, positioning the header bits for the next hop. With 32-bit flits, a packet can make a total of 15 hops [Bjerre05a]. Packets have variable length a control bit is used to indicate the last flit. The interface used to program the GS connections is implemented as an extension of the local port. Figure 2.21 shows the internal architecture of the BE router.

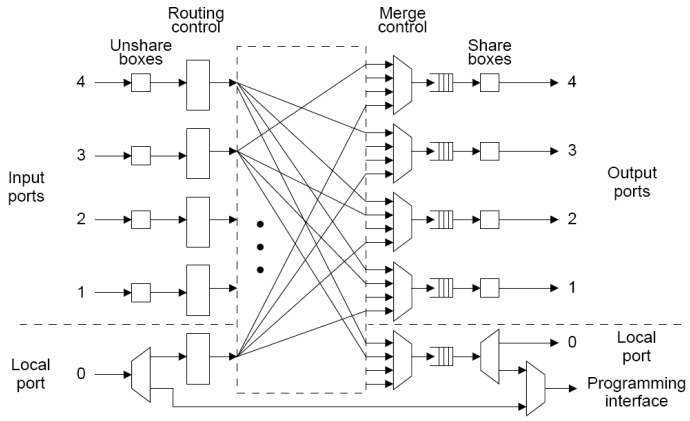


Figure 2.21 MANGO BE router [Bjerre05a]

The GS router is implemented as a non-blocking switching module. Each output port has seven GS communications and one BE communication. The GS communications are multiplexed using the virtual channel with a buffer per channel approach. These virtual channels are allocated as a circuit switching. Special BE packets are used to allocate and desallocate the GS virtual channels on the routers. Thus, GS channels behave as a circuit switching and GS packets do not need to carry the routing information. Figure 2.22 shows the BE router integrated into the GS router, using a subset of the VCs.

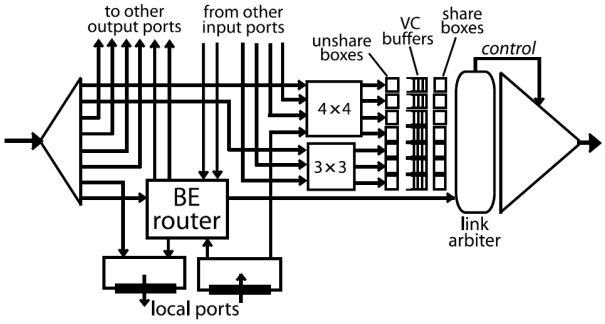


Figure 2.22 MANGO: BE router integrated into the GS router [Bjerre05a]

MANGO use flow control signaling for each VC between routers. Thus, end-to-end flow control signaling is not needed. The flow control is implemented using share-based VC control [Bjerre04]. When a new flit of VC*i* has been transferred by an output port, the *share box i* (Figure 2.22) become locked, not allowing further flits to pass. The flit passes across the output port, the inter-router wires, the input port, the switching module, and arrives to the *unshared boxe i*. The *unshared box* implements a latch, into which the flit is accepted. When the flit in turn leaves the *unshared box*, a

unlock control wire toggle. This *unlock* control wire arrives to *share box i*, admitting another flit into the output port.

A 33-bit MANGO router using 0.13μm CMOS technology from STMicroelectronics has been implemented. The performance in netlist simulations using worst-case timing parameters was 420 Mflits/s. The estimated area is 0.277 mm^2 [Bjerre05c].

2.7.2 Analysis

MANGO is an asynchronous NoC designed for message-passing programming model. Its architecture supports seven GS communications and one BE communication per output port. The GS router uses the virtual channel with a buffer per channel approach. The GS communications are allocated using a circuit switching approach. Firstly, the VC are reserved by BE packets. Secondly, the GS can use the reserved path. Finally, the VC is desallocated using BE packets. Therefore, no collision can exist between GS communications. In terms of VC multiplexing, a fair allocation policy is implemented for each output port. Moreover, the BE and GS traffics are completely split by different VCs. Consequently, it is possible to define hard constraints on the latency and on the through of the GS communications [Bjerre05b].

The architecture is suited to GALS as the router and the links are designed using asynchronous logic. The IP cores are connected to the MANGO router through a network adapter (NA) which performs synchronization between the clocked IP and the clockless network.

In terms of bandwidth, the MANGO router cannot deliver burst transactions because the *share-based VC* approach limits it. Initially, the bandwidth is limited by the fair allocation policy of the *link arbiter* (Figure 2.22). However, if just one GS communication is used, the *link arbiter* can always be allocated to the same VC. In this situation, the bandwidth of the GS communication is no longer limited by the allocation policy. It is limited by the cycle-time of *share-based VC* approach, which is a round trip between *share box*, inter-router wires (long wires), GS switching module, *unshared box*, inter-router wires (long wires), and back to the *share box*. Consequently, the maximum bandwidth of a GS communication is the inverse of this cycle-time.

For deep submicron technology, where the long wire delays became predominant, the pipelining of the inter-router wires (the long wires) does not solve the bandwidth limitation in MANGO. This comes from the fact that the cycle time of

share_box-to-unshared_box is not modified, because the flit flow-control mechanism is between routers, not a link level.

Finally, the area of the architecture is too expensive because it requires many multiplexers while the buffering memory per channel is low. The architecture requires more than 40 multiplexers of 4-input to 1-output.

2.8 Intel Tera-scale

Intel unveils in [Vang07] an 80-tile processor architecture organized as 10x8 2D mesh and interconnected by an NoC. The circuit contains 100 Million transistors on CMOS 65nm and has been tested up to 5.1-GHz.

2.8.1 Architecture

Each tile contains a processor element (PE) and a router as shown in Figure 2.23. The PE is a VLIW processor containing two independent fully-pipelined single-precision floating-point multiply-accumulator (FPMAC) with 3KB of instruction memory (IMEM) and 2KB of data memory (DMEM). Detailed description of FPMAC can be found in [Vang06].

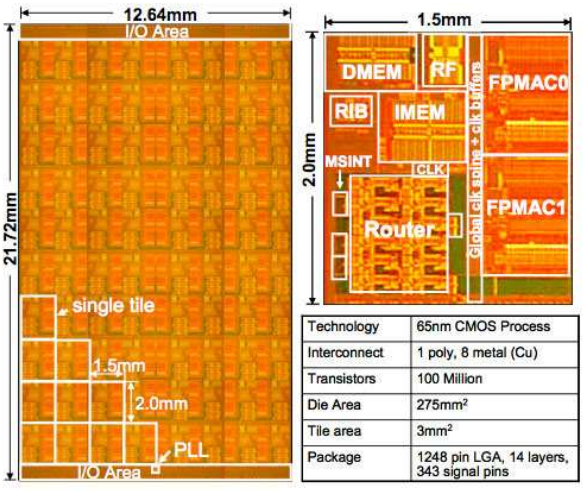


Figure 2.23 Tera-scale die micrograph [Vang07]

The router is a 5-port wormhole-switch with two logical lanes (virtual channels) for death-lock free routing, and a fully non-blocking crossbar switch with a total bandwidth of 80GB/s. The FIFO depth of each queue is 16 flits, and each queue has an arbiter and a flow control logic (Figure 2.24). The router uses 5-stages pipeline with two-stage round-robin arbitration scheme.

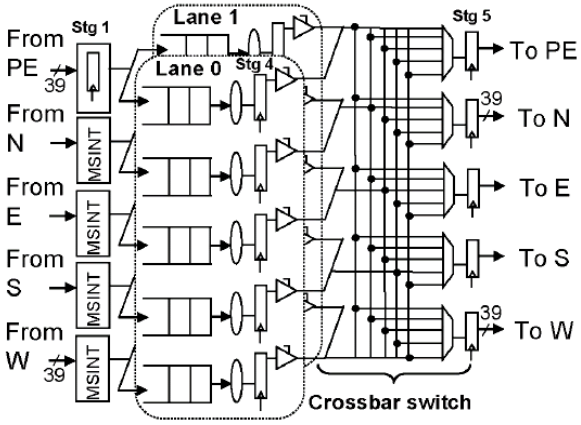


Figure 2.24 Tera-scale router [Vang07b]

The inter-router links are 39-bit unidirectional point-to-point links. Packets are subdivided into flits, each flit consisting of 32-bit data and 6-bit control signals (Figure 2.25). The packet header allows a 10-hop source routing path, where each hop is encoded in 3 bits. A chained headers (CH) bit in the packet header provides support for larger number of hops. Flow control and buffer management between routers are debit-based using almost-full bits, which the receiver queue signals via two flow control bits (FC), when its buffer reaches a specific threshold.

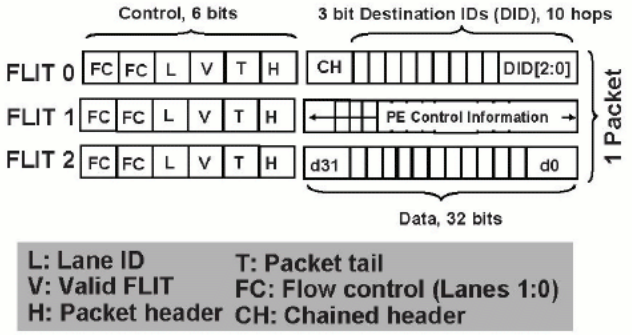


Figure 2.25 Tera-scale packet format [Vang07b]

The Tera-scale router architecture, which is described in [Vang07b][Vang05], was adapted from an off-chip network router described in [Wilso01]. Its architecture was simplified with dual edge-triggered flip-flops and a reduced number of logical lanes. Therefore, its area is 0.34mm² in 65nm technology.

The circuit uses a global mesochronous clocking. Each tile is synchronous while the communications between the tiles are mesochronous. The on-chip PLL output is distributed on a differential manner over horizontal and vertical spines on M7 and M8. An opamp at each tile converts the differential clock to a single-edge clock with 50% duty cycle as shown in Figure 2.26. Therefore, the intra-tile clock skew is 4ps

while the inter-tile clock skew can be around 200ps. The global clock distribution power at 4GHz, 1.2V supply is 2.2W. This clocking scheme dropped the typical clock distribution power from the typical 30% of the socket power to roughly 10% of the total socket power [Baut07].

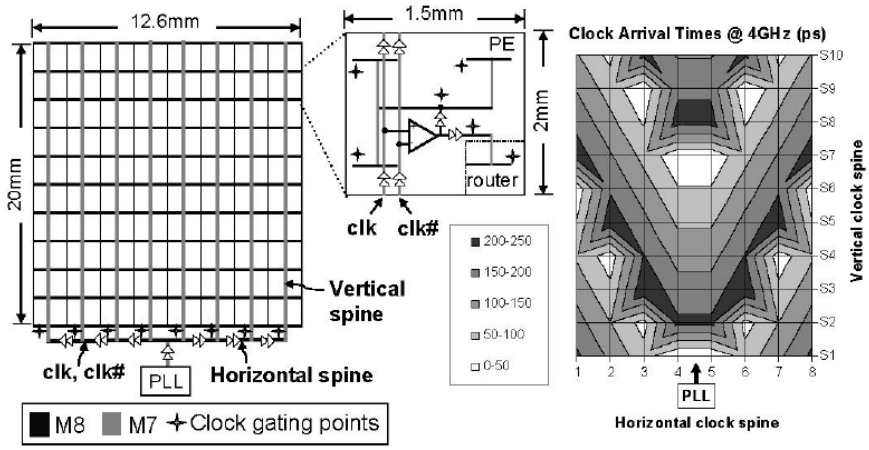


Figure 2.26 Tera-scale clock distribution [Vang07]

Mesochronous links are interconnected using clock-phase insensitive communications (MSINT). The MSINT architecture is a 4-word deep circular FIFO built using latches capturing data on both edges. This type of interface is analyzed in the Appendix A. Figure 2.27 shows the FIFO architecture and its timing diagram. A strobe signal (Tx_clk) is delayed using a programmable delay line in order to latch the data on the data-latches. A synchronizer circuit set the latency between the FIFO write and read pointers to 1-2 clock cycles based on the phase of the arriving strobe signal with respect to the local clock signal. A more aggressive low-latency setting reduces the synchronization penalty by one clock cycle.

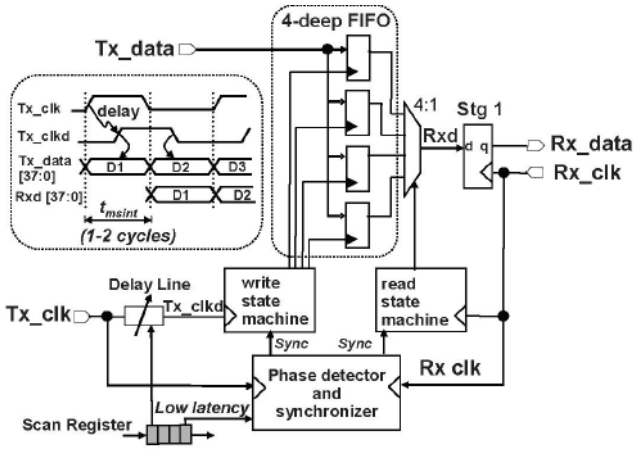


Figure 2.27 Tera-scale mesochronous interface [Vang07b]

2.8.2 Analysis

The Tera-scale router is a simplified and optimized version of an off-chip network router able to be clocked up to 5.1 GHz. The router contains two virtual channels, one for the request and another for the response packets. No guaranteed service traffic is detailed.

The router uses source routing algorithm, thus the network interface controllers (Router Interface Block RIB in Figure 2.23) have to be programmed in order to route the packets over the network. Moreover, the architecture is message-passing oriented due to the source-routing algorithm.

The physical implementation is custom, as the circuit requires sizing all the transistors of the design to achieve the required performances. Thus, no standard cell implementation is possible. Moreover, the router takes 0.34mm² in CMOS 65nm, which is several times bigger than targeted architecture of this thesis.

The Tera-scale mesochronous links (MSINT) uses a synchronous latency-insensitive design. This architecture is suited to interface mesochronous links. The latency of these interfaces has to be accounted for the packet router latency. Thus, the packet latency is 6-7 clock cycles (5 from the pipelined router and 1-2 from the MSINT). In terms of area, the MSINT interface can be estimated to 0.0112μm² from the die micrograph of Figure 2.23.

2.9 Conclusion

In this chapter, we have analyzed the most significant published Network-on-Chip architectures. The Æthereal, Nostrum, ANOC, QNoC, and MANGO architectures have guaranteed service traffic, and the SPIN, ANOC and Tera-scale architectures have been physically implemented on silicon.

The experience gained in the physical implementation of the 32 ports SPIN network was precious to define a new architecture, well suited to the Globally Asynchronous, Locally Synchronous (GALS) paradigm. The SPIN architecture was not suitable to be physically implemented with commercial tools. Therefore, we will target a fully synthesizable architecture using synchronous standard cells only, without either asynchronous or custom cells.

Asynchronous NoC seams to become popular as ANOC, QNoC, and MANGO have been designed following an asynchronous approach. However, the lack of commercial tools and the complexity to synthesize, verify, and test the implemented

circuit are strong limitations to the introduction of these architectures in commercial products.

In terms of number of traffic classes, the complexity of the router increases rapidly with the number of channels. The QNoC and MANGO architectures support 4 and 8 different traffic classes respectively. We believe that the actual requirement for guaranteed service traffic is no higher than two traffic classes: best effort and guaranteed service. Therefore, we prefer to improve the throughput of the router by increasing the FIFO depth rather than increasing the number of channels. From this point of view, the ANOC router is a good tradeoff as it implements two traffic classes.

The multiplexing of the traffic classes should be fair without starvation situations. The QNoC and ANOC architectures have a fix priority, thus provoking starvation on the low priority channels.

The allocation and reallocation of the GS communications should not require to program the complete network, as the Æthereal (distributed version) or MANGO NoCs. The allocation of the GS channels should be easy to modify dynamically.

In order to design a SoC compatible with the GALS approach, the NoC architecture should not require a global synchronicity. The Nostrum and Æthereal NoCs requires some sort of global synchronicity. This synchronicity constrains the Back-End implementation. We believe that a mesochronous clock distribution and a flow control at link level is a good tradeoff between global synchronicity and Back-End effort.

In terms of routing algorithm, we prefer an address-based algorithm rather than a source routing algorithm. The motivation is double. Firstly, the Network Interface Controller is simpler because the destination address can be used easily derived from the routing address. Secondly, the source routing limits the scalability, as it requires a path extension mechanism when the routing path does not fit into a single flit.

In terms of implementation strategy, we believe that a synchronous standard cell implementation flow with neither asynchronous nor custom cells is more suited and flexible for an industrial product. Optimized architectures such as the Tera-scale network, are suited to high performance computing but not for handheld or mobile phones. Its power consumption and its silicon area are too excessive for these applications.

In order to simplify the Back-End and to improve the portability, we believe that a soft macro implementation is preferable to the hard macro approach used by Æthereal and ANOC.

Chapter 3

Guaranteed Service

This chapter describes the implementation of the guaranteed service in the DSPIN architecture. The chapter starts with a statistical approach. As expected, this solution cannot guarantee a strictly bounded latency. Consequently, a virtual channels implementation was proposed and analyzed in order to obtain guaranteed service. The solution proposed is analyzed in terms of performance and implementation cost.

The contributions of this thesis to the DSPIN architecture were not limited to the implementation of the guaranteed service traffic. Some additional improvements were performed on the initial DSPIN architecture and they are summarized in this chapter.

The DSPIN architecture was simulated in SystemC and VHDL RTL in order to obtain the saturation threshold and to verify the hard constraints obtained on the guaranteed service packets. Moreover, a simulation platform was analyzed in order to characterize the performance of the network in function of the FIFO depths and the packet length.

Finally, the DSPIN architecture was synthesized and its performances are analyzed in terms of area and maximum clock frequency.

3.1 Statistical Guaranteed Service

This section describes a first study were the DSPIN architecture presented in the State of the Art chapter was used to route packets with two levels of priority, without using independent hardware resources.

3.1.1 Priority Allocation

In this implementation, the DSPIN architecture was modified to take into account a priority allocation of the priority packets. This work was started by Nicolas Guillermin (intern at The University of Pierre et Marie Curie) and later improved in this thesis.

In order to differentiate the *priority* packets from the *normal* ones, the DSPIN packet incorporated a priority flag on the first flit of the packet. The allocation priority of output ports was modified in order to allocate more often the *priority* packets than the *normal* ones. Moreover, the modified allocation priority was designed to guarantee no starvation situations. Thus, the *normal* packets have at least 1 chance in N to be granted. Consequently, the *priority* packets can flow on the network with higher priority than the *normal* ones. However, a *priority* packet cannot suspend a *normal* packet that is being served.

We simulated a 4x4-network to analyze the performances of the implementation. In order to simulate a request-response network, each node of the network contains a packet initiator (which chooses randomly its destination cluster) and a packet target. 12 of the nodes use *normal* traffic and the other 4 uses *priority* traffic. The offered load of the *normal* traffic is 80% while the offered load of the *priority* ones is 5%. The offered load is the ratio between the number of injected flits and the total number of cycles. Thus, the *priority* packets will try to flow on a saturated network. Figure 3.1 shows the probability distribution of the packet latency for the *priority* packets when the network uses the priority allocation (blue line) and when the priority allocation is disabled (green line).

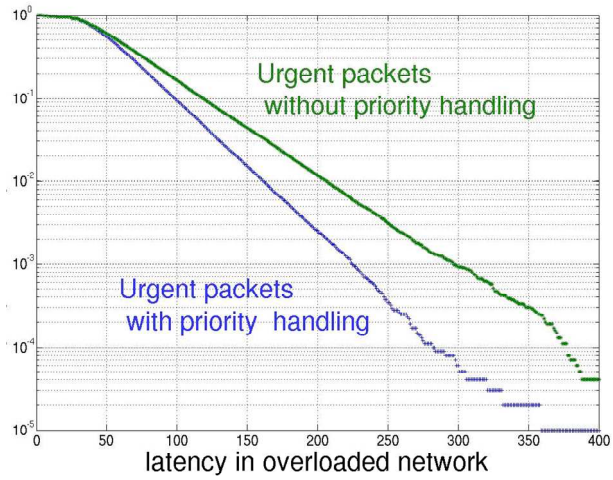


Figure 3.1 Probability distribution of the packet latency on an overloaded network

The priority allocation reduces the average latency of the *priority* packets from 75 to 65 clock cycles. However, the probability distribution, even when the priority allocation is enabled, shows a latency over 220 clock cycles for some packets (3 times higher than the average), which is not suited for real time applications.

In order to simulate complex SoC where the number of IP is higher than 32, a new platform of 10x10 clusters was built. This platform uses new traffic generators and targets, which send packets to the others clusters using non-uniform random distribution. This feature simulates the locality of actual embedded applications: a cluster communicates more often with its near neighbors rather than with its far neighbors. As example, Figure 3.2 shows the distribution of the destination cluster for the cluster (3,3). Brown color means frequent destination while blue color means infrequent destination. The cluster (3,3) is depicted in blue as the destination cluster cannot be the source cluster. The packets addressed to the same cluster are treated locally. Moreover, the length of the packets is a non-uniform value between 1 and 16 flits.

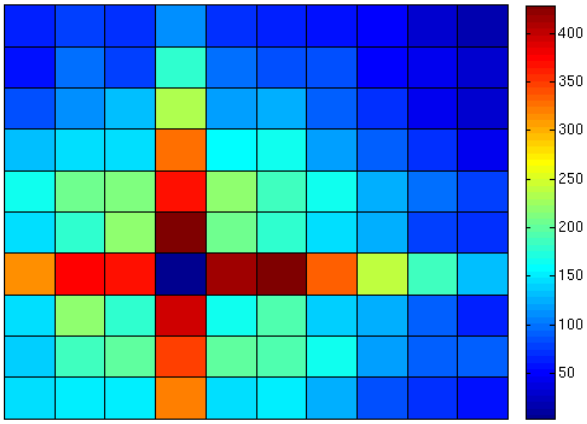


Figure 3.2 Distribution of the packet destination

Figure 3.3 shows the probability of the distribution of the packets latency of cluster (1,4) under two simulation conditions (priority and normal) and under two offered loads (20% and 30%). The improvement on the latency when the packets are sent in a priority way is notable. However, the latency of the *priority* and the *normal* packets is drastically reduced by reducing their offered load. However, at 20% offered load, the improvement of the *priority* packets latency is not enough to guarantee hard bounds on the latency.

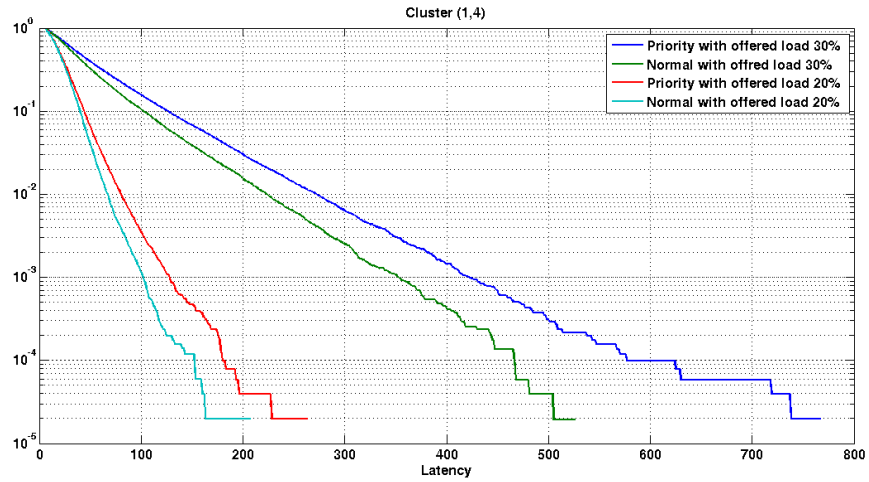


Figure 3.3 Probability distribution of the packet latency for 20% and 30% offered load

The priority allocation is a simple and low cost way to improve the latency of some priority packets. However, the guarantees in terms of latency are very soft (sadistically).

3.1.2 Priority Allocation with Suspended Low Priority Packets

This implementation is an improvement of the previous algorithm. The improvement is the ability to suspend a *low priority* packet to grant the resource to a *high priority* packet. A *low priority* packet can only be suspended by a *high priority* packet while a *high priority* packet cannot be suspended by neither *low* nor *high priority* packets. The allocation algorithm is more complex because it has to manage the priority conditions and a *suspend* state. Figure 3.4 shows the suspend mechanism where a *high priority* packet (red) temporally suspends a *low priority* packet (green).

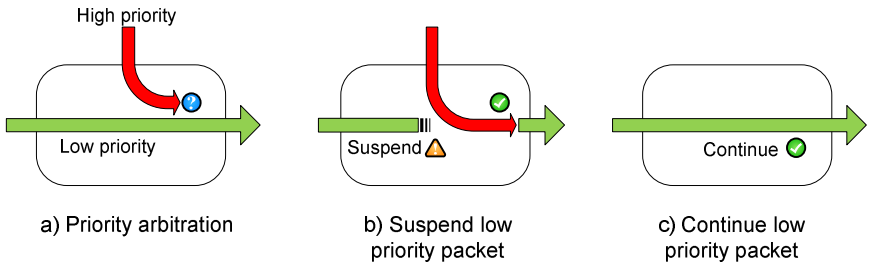


Figure 3.4 Suspend mechanism

The suspend algorithm is an inexpensive way to improve the latency of h*igh priority* packets. However, the suspend algorithm has some limitations and can induce deadlock situations.

- **Same destination**: A *low priority* packet should not be suspended by a *high priority* packet if both have the same destination cluster. Otherwise, the Network Interface Controller should be able to handle two interleaved packets. Consequently, the packets cannot be treated as an atomic transaction and the NIC becomes more complex.
- **Deadlock situations**: The combination of two factors can generate deadlock situation: Firstly, the *low priority* packets are stalled until the end of the *high priority* packet. Secondly, the packets cannot be interleaved by the IP. Example (Figure 3.5): a *low priority* packet (A) that is suspended by a *high priority* (B) one. The *high priority* packet (B) is waiting the end of a *low priority* packet (D) with the same destination cluster. However, the *low priority* packet (D) is also suspended by a *high priority* packet (C) that has the same destination of the first *low priority* packet (A). Therefore, a deadlock situation is generated, as the IP cannot interleave the packets.

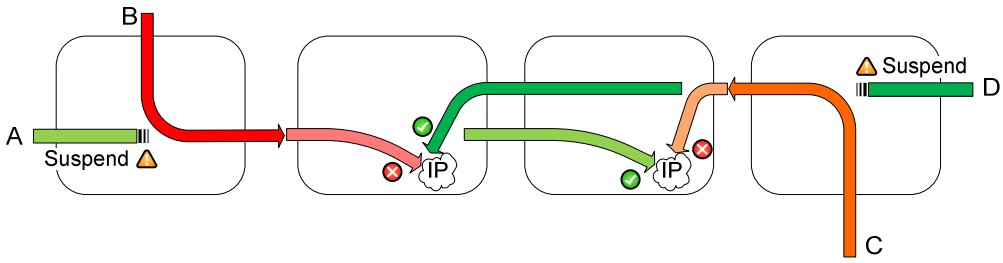


Figure 3.5 Deadlock on priority algorithm with suspended packets

Taking into consideration all these factors, we simulated a 10x10 platform of the previous implementation and avoided the deadlock situations. Figure 3.6 shows the probability distribution of the packet latency under two simulated offered loads (10% and 20%).

The latency of the *high priority* packets was improved. However, the reduction of the offered load on the network had higher impact on the reduction of the packet latency. The latency guarantees obtained with this algorithm are very soft, very similar when compared with the previous algorithm. In terms of complexity, the algorithm is not too complex; however, it is not deadlock free.

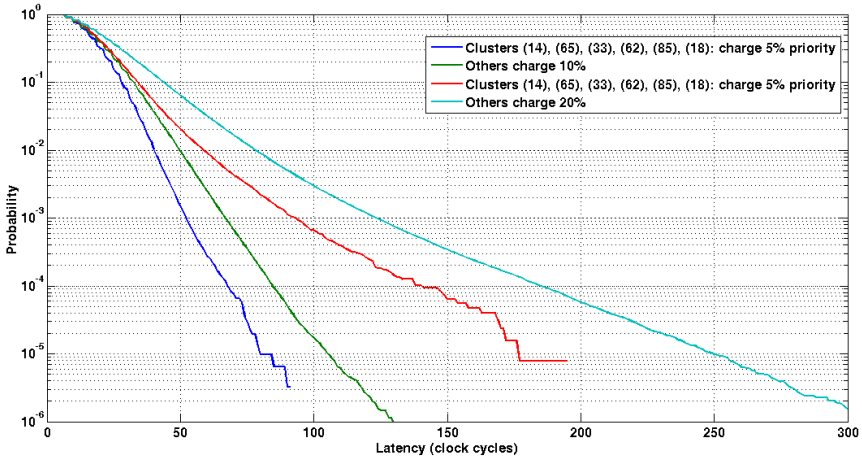


Figure 3.6 Probability distribution of the packet latency with the suspended mode

3.1.3 Statistical Guaranteed Service

We presented two implementation techniques to improve the latency of the priority packets. These algorithms obtain soft guarantees on the latency while its implementation is not very costly. However, the network load has higher influence on the latency of the packets rather than its traffic type. Consequently, it is better to control the accepted load of the *low priority* traffic rather than to arbitrate the traffic priorities inside the network.

Moreover, the suspend algorithm is not suited due to its deadlock situations and the implementation complexity of the Network Interface Controller. In order to avoid the deadlock situations without a complex NIC, the *low priority* packets should not be completely stalled until the end of the *high priority* packets. Consequently, we decided to investigate a virtual channel approach to implement the guaranteed service traffic.

3.2 DSPIN Architecture with Guaranteed Service

As seen in previous paragraph, the statistical guaranteed service traffic does not meet the bounded latency guarantees addressed by real time applications. In order to provide Guaranteed Service traffic in the DSPIN architecture we use the Virtual Channel (VC) technique with a buffer per virtual channel. Thus, logically independent channels share the same physical channel. The advantage of this technique compared to the priority allocation of the previous section, is a full separation treatment of the traffic classes. Thus, when one traffic class is blocked the

other is neither suspended nor blocked. Consequently, the deadlocks situations of the *priority allocation with suspend low priority packet* (analyzed in previous section) can be avoided.

The main advantages of the virtual channels technique is a low area overhead per additional virtual channel and a reduced wire congestion of the long wires. This stems from the fact that the traffic classes are multiplexed over the same long wires. Moreover, we chose the VC technique with a buffer per virtual channel and not the VC with a buffer per link in order to avoid the need of a full network synchronicity (see Chapter 1) and minimize the complexity of the NIC and router.

We decided to use two virtual channels per router port. Thus, two traffic classes can be classified, Best Effort (BE) and Guaranteed Service (GS) packets. Fort both traffic classes, the router use the same routing and switching algorithm, and the same packet format. A traffic is considered GS traffic because it is sent on a GS port, otherwise it is a BE traffic. Moreover, a GS traffic enters the network through a GS input port, travels the network over GS FIFOs and exits the network through a GS output ports, vice versa for the BE traffic. The GS and BE traffic share the VC links but not the storage elements.

3.2.1 DSPIN Router

The virtual channel implementation has been largely used in wide-area networks to multiplex different traffic classes over the expensive resources, which are the inter-router wires (Figure 3.7a). In DSPIN architecture, the costliest resources are actually the intra-cluster long wires. Therefore, the virtual channel is implemented inside the router and not between routers (Figure 3.7b). The virtual channel interconnects the modules of a DSPIN router. Thus, the inter-cluster communications use point-to-point physical links while the intra-cluster communications use the virtual channel. The advantage of this technique is that the virtual channel is embedded inside the cluster, which is an isochronous island. Therefore, the timing closure effort is simplified, and the cluster can be implemented as a synchronous stand-alone entity and finally assembled with other clusters. The uncertainty of the inter-cluster wires is low because these wires can be very short as the clusters can be placed side by side and the router modules of different clusters can be aligned.

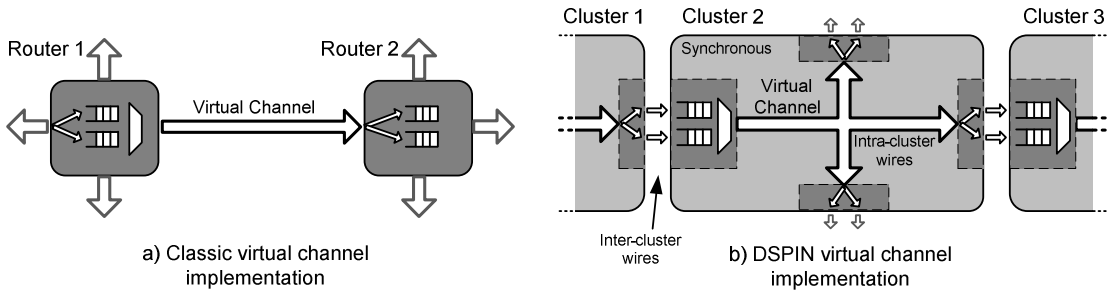


Figure 3.7 Virtual channel implementation

The implementation of the Virtual Channel requires independent storage elements for independent traffic classes. Therefore, independent FIFOs are used per GS and BE traffic. Figure 3.8 shows the DSPIN architecture and the GS and BE FIFOs. Compared to the initial DSPIN architecture (see Chapter 1), the number of FIFOs is doubled while preserving the same number of long links, because over these links both traffic classes are multiplexed.

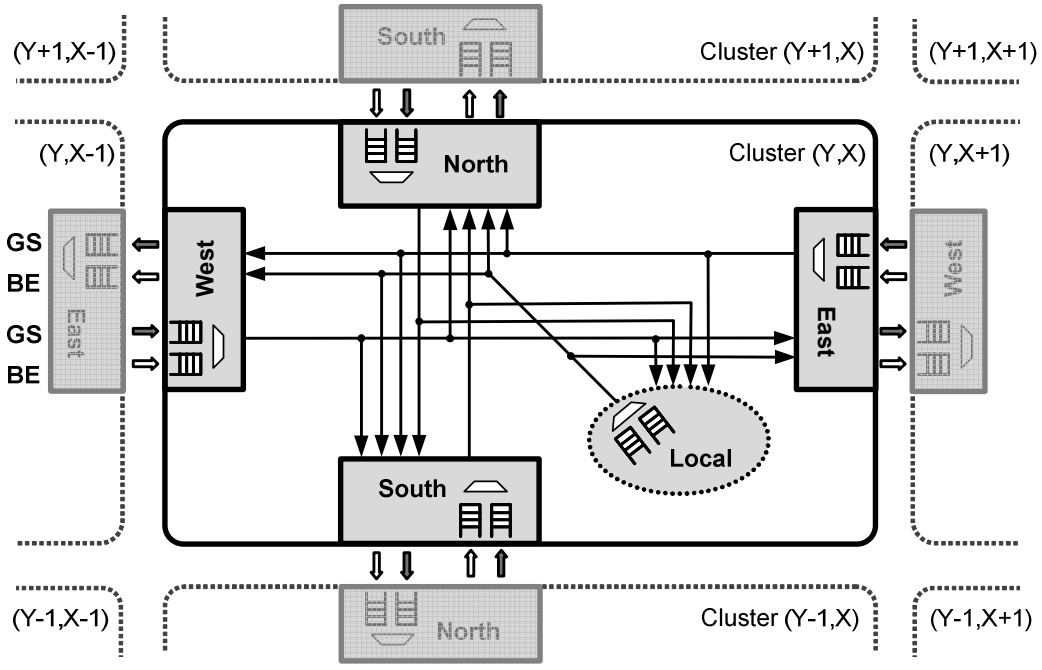


Figure 3.8 DSPIN router architecture

3.2.1.1 DSPIN router modules

The DSPIN router contains four modules placed on the North, South, East, and West sides of the cluster, and a Local module placed inside the cluster. Each module contains two bi-synchronous FIFOs, two address decoders, three multiplexers, and three state machines as seen in Figure 3.9. A module can be decomposed into two submodules the *sender* and the *receiver*, top and bottom submodules on Figure 3.9

respectively. On both submodules, the BE and GS interfaces are split. A packet enters the router through an input port of a *receiver* submodule, it is routed between the submodules, and it exits the router through its corresponding output port of a *sender* submodule. The *receiver* submodule treats the packets as follows:

- The BE and GS packets arrive through the BE and GS input ports.
- The packet flits are stored into bi-synchronous FIFOs.
- The packets are decoded using independent packet decoders. If the *begin_of_packet* bit is set on the flit, the destination address is analyzed using the routing algorithm. Thus, a request signal (Req) is sent to the corresponding *sender* submodule to request the packet transmission. Independent request signals are sent for each *sender* submodule and for each channel (BE and GS).
- The TDM state machine controls the allocation of the *in* multiplexer, which multiplexes the BE or GS flits over the virtual channel. The detail description of the TDM state machine is described hereinafter.
- The virtual channel wires contain the FIFO flit data, a valid data bit, and a TDM bit. This last bit identifies the traffic class (BE or GS).
- When the data is correctly transferred, the *sender* submodule responds with an acknowledge signal (Ack), which is used to dequeue data from the corresponding bi-synchronous FIFO.

The *sender* submodule treats the request from the *receiver* submodules following the next steps:

- Two state machines (*BE* and *GS*) treat the request received from the *sender* submodule. The BE state machine on submodule *i*, only treats the BE request which are addressed to submodule *i*, and vice versa for the GS state machine.
- The *BE* and *GS* state machines control the allocation of the output ports. The output ports can be allocated to one of the virtual channels or can be invalidated (*no_allocated* state) when no data is routed. The detailed description these state machines is described hereinafter.
- The allocation algorithm works as follows:
    - If the state machine is in the *no_allocated* state and a new request arrives, the output port is then allocated to satisfy the request.
    - If the state machine is allocated to treat a request and a second request arrives, the second request waits until the end of the first

one to be satisfied. In case of multiple waiting requests, a round robin allocation policy is used to satisfy equally the requests.

- The end of a request is detected with the reception of a flit with the *end_of_packet* bit asserted. Therefore, the state machine toggles to a waiting request or to the *no_allocated* state if none is waiting.

- When a data is transferred to an output port, an acknowledge signal (Ack) is sent to the corresponding *receiver* submodule.

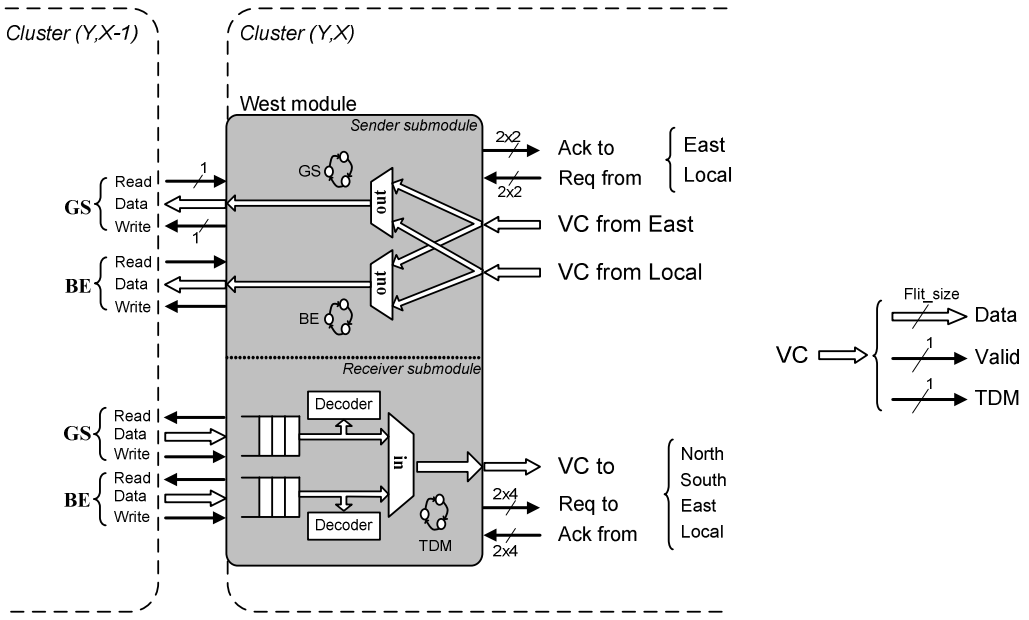


Figure 3.9 West module router detail

3.2.1.2 TDM state machine

The TDM state machine guarantees, by construction, that both traffic classes have access to the Virtual Channel even when a traffic class is blocked. The TDM is implemented as a Moore state machine. The simplest allocation policy is round robin, where each traffic class has a guaranteed 50% of the bandwidth (Figure 3.10a). It can be modified in order to give more guaranteed routing slots to the GS traffic (Figure 3.10b). The round robin algorithm guarantees equality between the BE and GS traffic while the modified algorithm, gives up to N times more guaranteed slots to GS traffic than to the BE traffic.

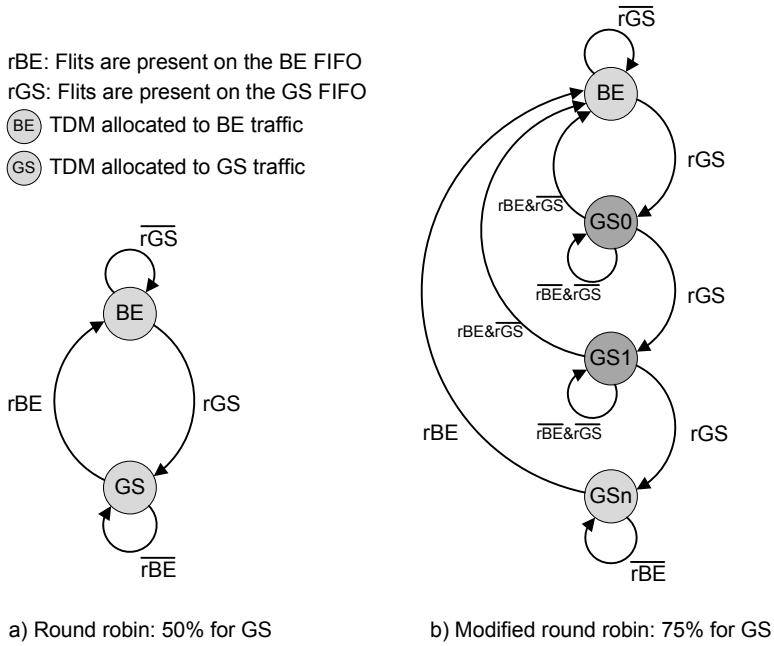


Figure 3.10 TDM state machine

Both algorithms are starvation free with maximized throughput. The throughput of the VC is maximized because when no BE traffic requires the VC, the state machine remains allocated to the GS traffic. Thus, all the VC slots are allocated to the GS traffic. The same phenomenon occur with the BE traffic when no GS traffic requires the VC.

3.2.1.3 Allocation state machines

The BE and GS state machines are identical. They are Moore state machines with round robin allocation policy to guarantee no starvation situations. Figure 3.11 shows the BE/GS state machine on the North module. This state machine has four allocated states (South, East, West, and Local), and a *no_allocated* state. When the state machine is in an allocated state (Allocated_to_East for example), the output port is allocated to the corresponding virtual channel (East virtual channel for example). When the state machine is on the *no_alloacted* state, the output port is invalidated as no packet is routed. This state is used to invalidate the output data and to reduce the power consumption of the router (see the clock gating section in Chapter 5 for further details) when no packet is routed.

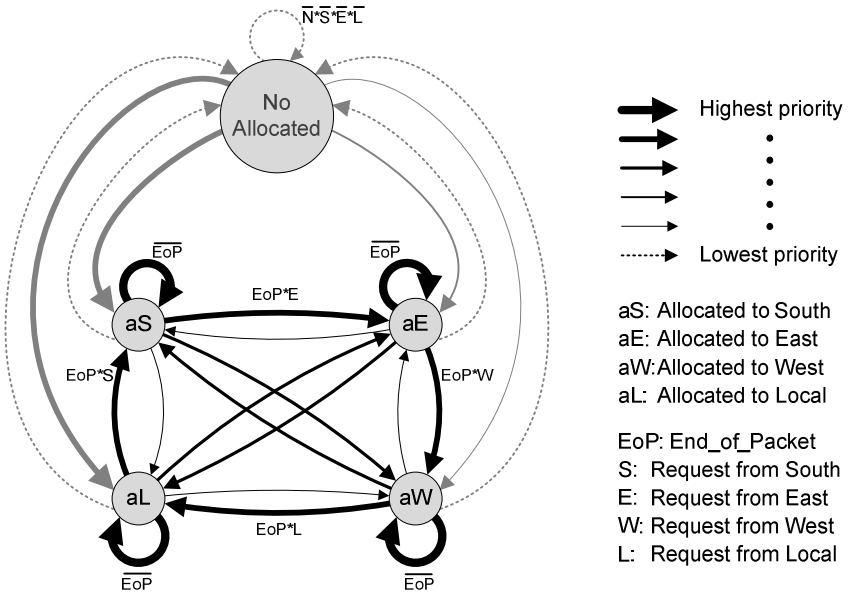


Figure 3.11 BE/GS state machine on the North module

When the state machine is allocated to one of the virtual channels (South for example) it remains allocated to that channel until the reception of the End_of_Packet (EoP). Then, the state machine changes its state using the round robin allocation policy. If no request is pending, the state machine switches to the *no_allocated* state.

The BE/GS state machines for the East and West modules are easier because the X-first routing algorithm does not allow a packet traveling on the Y coordinate to be routed to the X coordinate. For example, the packets coming from North module can be routed to neither East nor West modules. Therefore, the BE/GS state machines on the East and West modules have just two allocated states and a *no_allocated* state.

3.2.1.4 Routing Guaranteed Service Packets

The DSPIN architecture before this thesis used the X-first routing algorithm to route the packets over the network. The same routing algorithm was used for both request and response networks. However, the guaranteed service traffic requires modifying the routing algorithm of the response-packets to maximize the utilization of the network. This phenomenon is depicted in Figure 3.12. A-to-A' and B-to-B' are two independent GS communication and their paths do not conflict. However, the responses to theirs requests (A'-to-A and B'-to-B) conflict. To avoid packet conflicts on the response network, the response network use the Y-First routing algorithm (Figure 3.12b). Consequently, non-conflicting guaranteed service request-packets will always have non-conflicting response-packets. DSPIN uses X-First routing algorithm on the request routers and Y-First routing algorithm on the response routers.

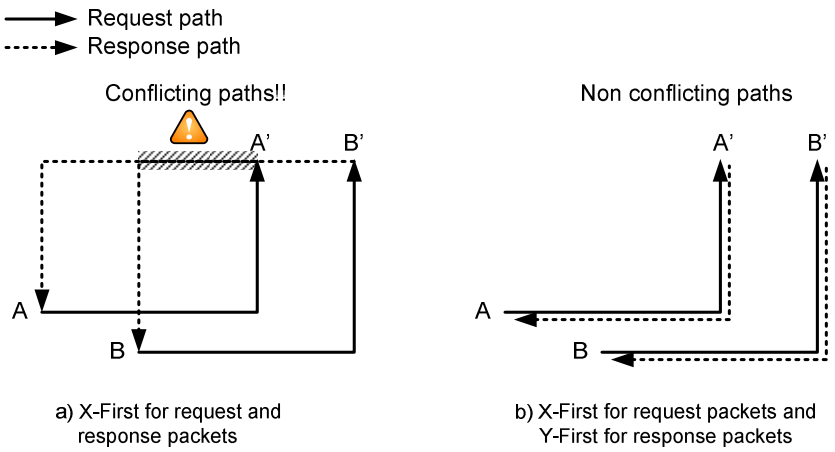


Figure 3.12 Request and response path analysis

3.2.2 DSPIN Network Interface Controller

The DSPIN Network Interface Controller interconnects the request and response routers to the local sub-system. The NIC provides services at the transport layer on the ISO-OSI reference model, offering to the local sub-system independency versus the network implementation. The actual implementation is compatible with the VCI/OCP [VCI00] protocol, but it can be easily adapted to any shared memory and transaction-based protocol.

Transaction-based protocols are composed of initiator IPs and target IPs. Initiator IPs issue request packets while target IPs return responses. The DSPIN NIC being a bi-directional bridge, behaves as an initiator and as a target as shown in Figure 3.13a. DSPIN NIC is also suited for hierarchical architectures as the one in Figure 3.13b. A local interconnect between the NIC and the IP can be used to split the intra-cluster communications from the inter-cluster communications.

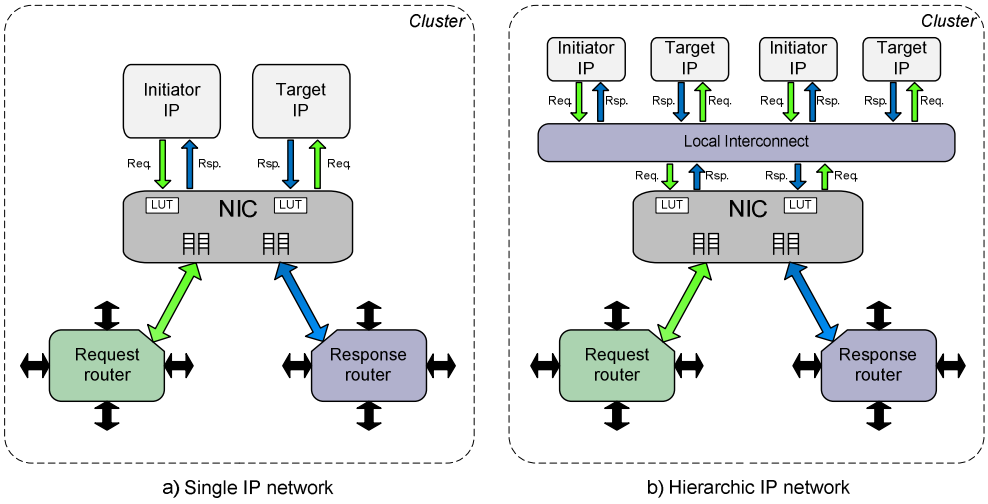


Figure 3.13 Network Interface Controller

The main tasks of the NIC are protocol conversion and packet building. The VCI/OCP protocol of the IP is implemented on the IP ports of the NIC while the Network ports of the NIC implements the DSPIN protocol. The protocol conversion algorithm differs for the IP initiator and IP target ports:

- **Initiator IP ports**: The IPs connected on the initiator ports send requests and receive the responses to their requests.
    - **Send a request**: The request packets are analyzed to identify the destination address and packet control signals. The destination address is translated to the DSPIN packet destination (Y,X). This can be directly done by taking the MSB bits of the IP packet address or by translating these MSB bits with a Look Up Table (LUT). The latter case gives more flexibility on the mapping addresses. Moreover, the packet control signals are analyzed in order to minimize the DSPIN packet length. For example, burst write requests packets with consecutive addresses are compressed by sending the beginning address and incrementing it on the destination NIC.
    - **Receive a response to a request**: The response packet to a request can be a read data, an acknowledge packet, or a fail packet. In case of read data, the IP data protocol is restored with the read data. An acknowledge packet can be the acknowledge response to a write request while a fail response can be an unmapped send request or a wrong operation request.
- **Target IP ports**: The IPs connected to target ports receive requests from the initiator IPs and send responses to these requests.
    - **Receive a request**: The protocol conversion restores to the IP the same information sent by the initiator IP. If the packet addresses were compressed due to burst write requests, they are restored using the beginning address and an incremental counter.
    - **Send a response to a request**: The response to a request can be the read data, acknowledge to a request or a fail request. In case of data, it is directly converted to a DSPIN packet. Acknowledge and fail requests are compressed in order to reduce the DSPIN packet length as they do not contain data bits.

Depending on management of the guaranteed service traffic by the IPs, two implementations of the NIC were designed for independent and mixed packet

treatment. On the one hand, when independent GS and BE IPs are used, a double channel NIC is designed. On the other hand, if the same IPs manage BE and GS traffic, a simple channel NIC is proposed.

3.2.2.1 Double channel Network Interface Controller

Double channel NIC have independent IP ports for the BE and GS channels. Hence, the BE and GS packets do not share the same interface (Figure 3.14). Thus, no deadlock situations exist on the NIC as BE and GS packets can continue to flow even when one of the traffics is blocked.

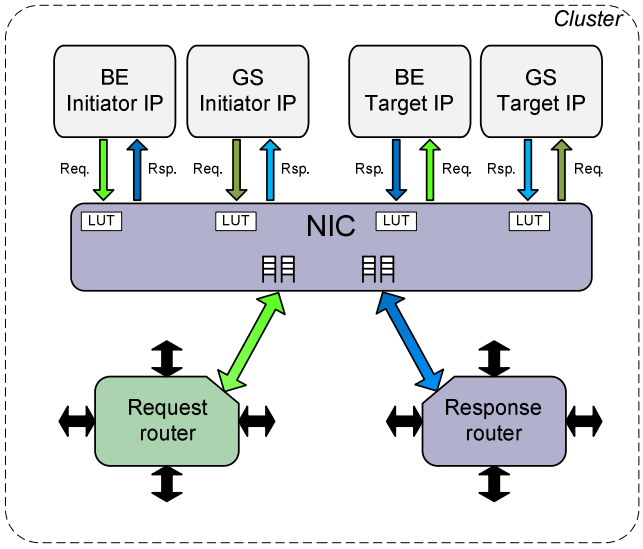


Figure 3.14 Double channel Network Interface Controller

3.2.2.2 Simple channel Network Interface Controller

A simple channel NIC combines the BE and GS packets on the same IP port of the NIC. Therefore, the IPs have to manage BE and GS packets. In order to differentiate the BE packets from the GS ones, two solutions are proposed: a flag-bit on the IP protocol or an identification by the destination address. For the latter, the destination address is analyzed to identify the packet type (BE or GS). This differentiation is performed using a Look Up Table (LUT-GS) which is implemented on the NIC, as shown in Figure 3.15.

3.2.3 Globally Asynchronous Locally Synchronous

In order to follow the GALS approach each cluster can have its own clock frequency without any frequency/phase relationship between its neighbors. In the DSPIN architecture before this thesis, the cluster clock frequency was used to clock the DSPIN router. Under these circumstances, a cluster clocked with a low clock

frequency will reduce the throughput of the network and increase the end-to-end path latency.

In order to split the cluster clock from the router clock, the DSPIN routers are clocked with an independent clock frequency (CLK_noc). Therefore, the latency of the packets is now independent from the cluster's clock frequency.

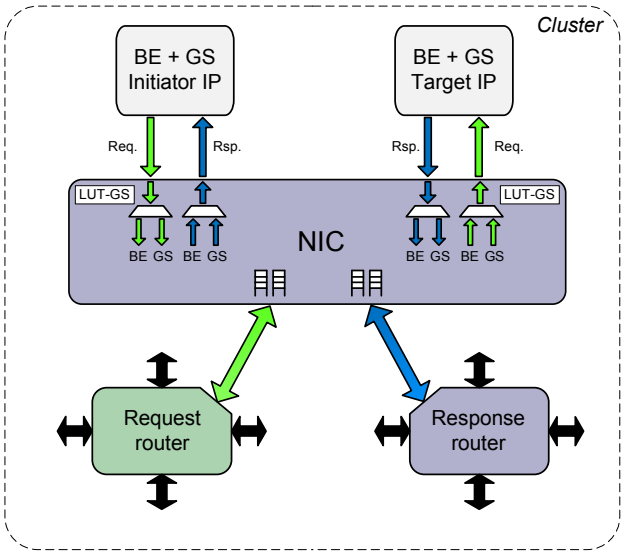


Figure 3.15 Single channel Network Interface Controller

In terms of clock distribution, a fully synchronous NoC is contradictory to the GALS approach. Therefore, we decided to distribute a mesochronous clock signal over the circuit, which is less complex to build, and less power consuming than a fully synchronous clock distribution. Consequently, all the routers are clocked with the same clock frequency but a clock skew can exist between neighbor routers.

The mesochronous communications between neighbor routers are carried out by bi-synchronous FIFOs (in mesochronous mode) [Miro07b]. To avoid the metastability issues, these FIFOs require that clock rising edges of the producer and consumer clock sides are not too close from each other (see Chapter 4 for further details). In order to separate the clock rising edges, we decided to invert the clock signals between neighbor routers as shown in Figure 3.16. Thus, the rising edges of the clock signals are 180° out of phase. Moreover, the FIFO continues to be operational even when the phase shifts between ±90° due to clock skew of the mesochronous clock tree. The methodology involve adding a clock inverter on the (Y,X) router where X+Y is an even number (Figure 3.16). Thus, neighbor routers have inverted clock signals. Chapter 5 details the physical implementation methodology of these mesochronous clock trees.

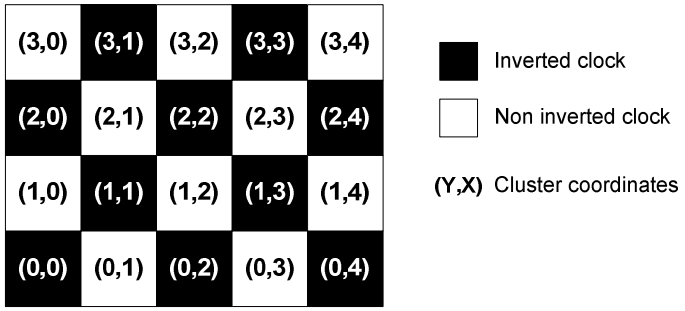


Figure 3.16. Inverted clocks signals on DSPIN routers

The communications between the router and the NIC are asynchronous because the subsystems can use independent clock frequencies. Theses communications are carried out by bi-synchronous FIFOs (in the asynchronous mode). Compared to the previous bi-synchronous FIFOs, these FIFOs can interface mesochronous and asynchronous communications, but with a higher latency (see Chapter 4 for further details).

3.2.4 Predictability

Predictability is one of the major issues in the design of a real time application. DSPIN guarantees, by construction, the predictability of the guaranteed service packets in terms of latency and throughput.

3.2.4.1 Guaranteed service path allocator

We have described until now how to handle two separated traffics on the same switching hardware. In order to guarantee an upper bound for the latency, and a lower bound for the throughput in the GS sub-network, we must guarantee that collisions in the GS sub-network will never happen (i.e. two different GS communication channels using the same path will not be simultaneously allocated). This requires some sort of end-to-end resource reservation (circuit switching). Following the Amdahl law, we do not want to pay hardware for un-frequent cases, and the end-to-end GS channel allocator is not implemented in hardware. For most embedded applications, the communication scheme is well known, and the system designer can statically allocate the required (non conflicting) GS channels. If static allocation is not possible, a GS channel allocator is implemented as a software task that will manage a global table of all existing GS paths, and perform dynamic allocation as required by the embedded software application.

3.2.4.2 Guaranteed service packet latency

The latency of the guaranteed service traffic is deterministic when no collision exists in the GS sub-network. In this section, we compute the upper bound of the GS packet latency under these circumstances.

On the one hand, the latency of the DSPIN router is predictable. Because, the DSPIN mesochronous clock distribution makes the network latency independent from the clock frequencies of the clusters.

On the other hand, no conflict between BE and GS packets exists inside the network, because the BE and GS traffics are routed independently, thanks to the virtual channel implementation.

- DSPIN uses independent virtual channels for *best effort* and *guaranteed service* traffics.
- Independent storage FIFOs are used for each virtual channel. The BE and GS packets share only the inter-cluster wires.
- The NIC has independent ports for *best effort* and *guaranteed service* packets.
- DSPIN uses round robin state machines without starvation situations.
- The TDM state machine guarantees no starvation situations and a maximum waiting time of one clock cycle before allocating a new *guaranteed service* packet.

Both the router predictability and the traffic independency make the routed traffic predictable. Taking into consideration the latency of the bi-synchronous FIFO (see Chapter 4 for further details), it is possible to compute the end-to-end path latency of the network. The bi-synchronous FIFO latency is 1.5 clock cycles in mesochronous mode and 2.5 clock cycles in asynchronous mode.

The packet latency is the end-to-end delay between the time a packet header enters into the network and the time it exits the network, assuming no contention. This path can be decomposed in three parts: First, Intermediate, and Last latencies. The First latency is the time it takes the packet to cross the first router. The Last latency is the time it takes the packet to cross the last router and the FIFOs on the NIC. The Intermediate latency is the time it takes the packet to cross an intermediate router between the first and the last router as shown in Figure 3.17. The latency of the state machines and FIFOs is expressed in function of the clock cycle (T). The first and last latency crosses a bi-synchronous latency with asynchronous mode, while the intermediate latency crosses a bi-synchronous FIFO in mesochronous mode. Thus, the intermediate latency is lower than the one of the first and last latencies. The BE,

GS, and TDM are Moore state machine with a latency of one clock cycle. In a general case, the packet latency is not elongated by the TDM state machine latency because its response time is hidden by the GS and BE state machines latency (as TDM and BE/GS are concurrent state machines). However, the TDM state machine can elongate the packet latency when the router has finished to serve a GS packet, no BE packet was waiting, and a GS and a BE packet arrive into the router at the same time. In this case, the GS packet has to wait one clock cycle. This phenomenon, which is rare, is due to the fair allocation of the TDM state machine.

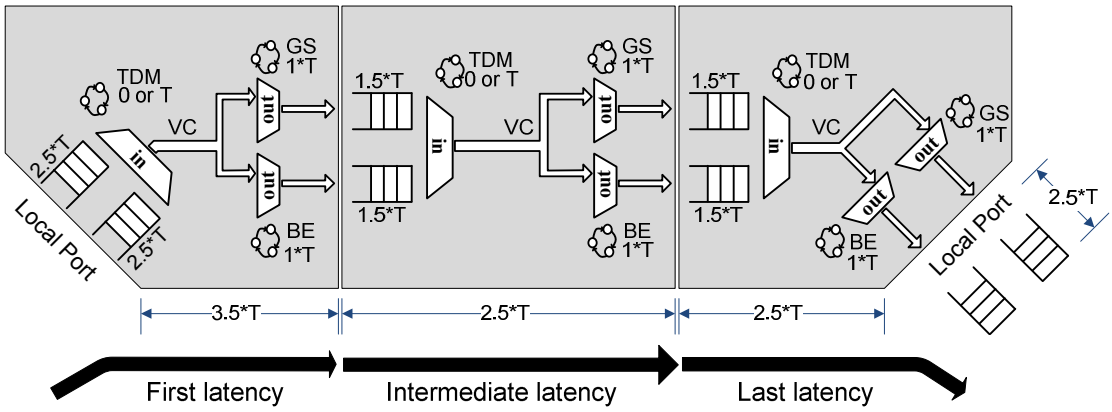


Figure 3.17 First, intermediate, and last router latency

Table 3.1 shows the packet latency on DSPIN network. In order to compute the hard bounds for the latency (maximum and minimum), the TDM state machine latency is considered. As an example, the total latency to cross 5 routers in a typical condition is (3.5 + 2.5*3 + 5)*T = 16 clock cycles (T). Its maximum latency is (4.5 + 3.5*3 + 6)*T = 21 clock cycles.

Table 3.1 Packet latency on DSPIN router

	Minimum (typical) latency	Maximum latency
First latency	3.5 * T	4.5 * T
Intermediate latency	2.5 * T	3.5 * T
Last router + FIFO	5.0 * T	6.0 * T

3.2.4.3 Guaranteed service throughput

The guaranteed throughput in DSPIN network is obtained thanks to a fair allocation policy of the TDM state machine. This state machine multiplexes the BE and GS traffics over the virtual channel. Thus, the bandwidth of the virtual channel is shared between BE and GS traffic. The maximum bandwidth of the virtual channel is

one flit per clock cycle. Assuming the round robin TDM state machine of Figure 3.10a, the bandwidth is equally shared between BE and GS. Therefore, the guaranteed throughput per GS traffic is 50% (one flit every two clock cycles). With the modified state machine of Figure 3.10b, the number of slots for the GS traffic can be increased to 75% of the total bandwidth (3 slots for GS and 1 for BE).

These GS bandwidths are guaranteed even if the BE sub-network is saturated. In case of not BE traffic using the same virtual channel, the TDM state machine allocates all the VC slots to the GS traffic. Thus, the GS traffic can potentially achieve 100% of the total bandwidth.

3.3 DSPIN summary

This section intends to summarize the DSPIN architecture, and report the contribution of this thesis [Miro06]. Some improvements on the DSPIN architecture are not covered by the guaranteed service section and they are mentioned in the next list. The lines marked with ❖ are a contribution of this thesis, the others have not been changed.

- DSPIN is a packet-based network on chip.
- 2D mesh topology.
- ❖ Flit size is generic. The flit contains two control bits (BOP and EOP). The routing address is contained in the first flit (8 bits) as shown in Figure 3.18). The *Error* and *Parity* bits have been removed from the flit control bits. They can be sent on the payload bits of the flit if necessary.
- ❖ Deterministic routing algorithm. The request routers use X-First while the response routers use the Y-First routing algorithm.
- DSPIN has best effort traffic.
- ❖ DSPIN has guaranteed service traffic. Hard bounds for the latency and the throughput can be guaranteed.
- ❖ DSPIN router has two versions, one with Best Effort (BE) and one with Best Effort and Guaranteed Service (BE + GS).
- DSPIN can be used for shared memory applications. It requires one router for request packets and one router for response packets.
- ❖ DSPIN can be used for message-passing implementation. Just one router per subsystem is needed.
- GALS compatible architecture.

- ❖ All the routers use the same clock frequency, but a clock skew can exist between routers (mesochronous). Neighbor routers have inverted clock signal to simplify the mesochronous interface.
- ❖ The subsystems can use an independent clock frequency and communicate asynchronously to the network.
- DSPIN architecture is distributed in 5 modules which are placed on the sides of the cluster.
- ❖ DSPIN is synthesizable with standard cells. Neither custom cells nor asynchronous cells are used.
- ❖ A power reduction mechanism is implemented using clock gating.

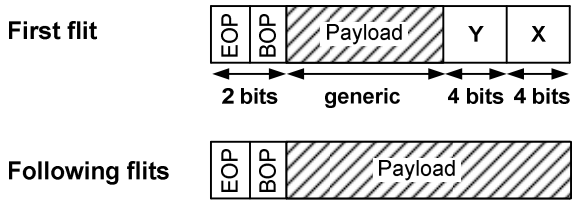


Figure 3.18 DSPIN packet format

3.4 Experimental Results

In this section, the DSPIN architecture is implemented on a simulation platform to verify the hard bounds of packet latency and the guaranteed throughput for the GS traffic. Moreover, a simulation platform is used to obtain the saturation threshold of the BE sub-network and to dimension the FIFO depth to maximize the network performance while preserving a small area. Finally, the DSPIN router is synthesized and its area and maximum frequency are analyzed.

3.4.1 Implementation Models

The DSPIN simulation models are part of the SoCLib [SOCLIB] project, which is an open platform for modeling and simulating multi-processors systems. The simulating environment is SystemC and the simulating levels are CABA (Cycle Accurate, Bit Accurate) and TLM/T (Transaction Level Modeling with Time).

DSPIN was implemented using SystemC at CABA level and later ported to VHDL RTL for synthesis. Both implementation models are compatible and can be exchanged in a SystemC/VHDL cosimulation environment. Thus, the simulation of a huge system can be accelerated by replacing the VHDL model with the SystemC model.

3.4.2 Simulating

A simulation platform was designed to evaluate the system performances. This simulation platform has a 10x10-cluster topology with separate request and response networks. The depth of the BE and GS FIFOs are 8 and 4 flits respectively. Each cluster contains one BE initiator, one BE target, one GS target, and one optional GS initiator. The average packet latency is measured as the average number of cycles for a round trip from an initiator to a target, and back to the same initiator. For each initiator, the offered load is the ratio between the number of injected flits and the total number of clock cycles. The BE traffic has a uniform random distribution (each BE initiator randomly sends packets to all BE targets). The packet length is a random value between 1 and 16 flits. If we plot the average latency versus the BE offered load (Figure 3.19), we see a saturation threshold of 25% for the BE traffic (blue line) while the latency of the GS communications (green and read lines) are not modified. In case of saturation, part of the BE offered load is not accepted by the network, but the GS traffic is clearly not impacted by the BE traffic. The latency and throughput of the GS traffic have been analyzed. For example, the latency of the network for the roundtrip between cluster (8,9) and cluster (5,3) is deterministic and equal to 62 cycles.

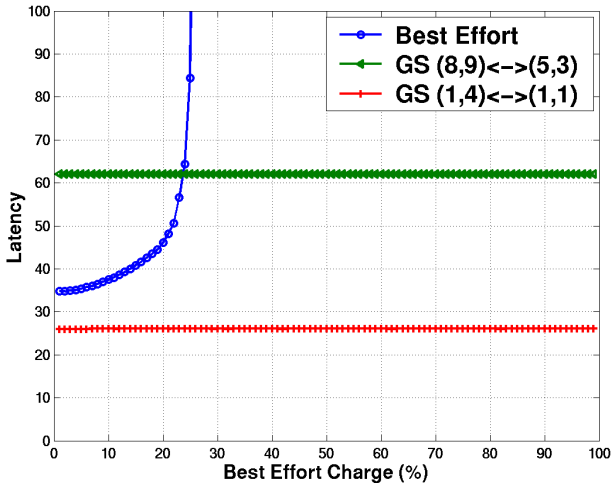


Figure 3.19 BE and GS latency in fucntion BE offered load

The throughput of each GS channel is guaranteed up to 50%, due to the round-robin allocation of the TDM slots. Figure 3.20 shows the GS latency in function of the GS offered load for a saturated and non-saturated BE sub-network. The throughput of the GS traffic is guaranteed up to 50%, even when the BE sub-network is saturated. On a 500MHz implementation, each GS channel has a guaranteed bandwidth of 8 Gbps.

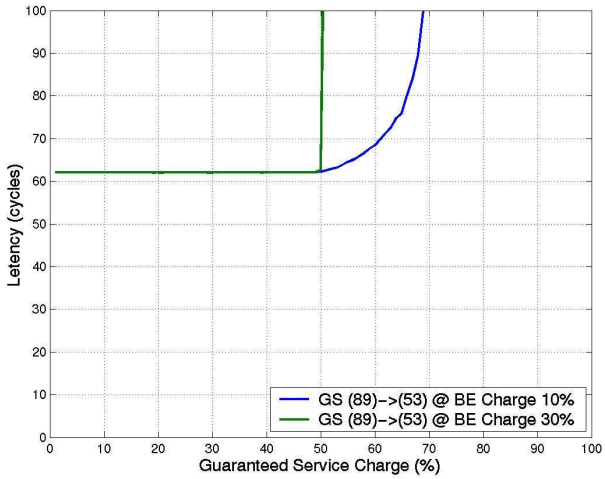


Figure 3.20 GS latency in function of GS offered load

The performances of the BE packets are determined principally by four factors: the FIFO depth, the packet length, the number of routers between the sender and the receiver, and the network load. The impacts of these parameters are analyzed in the next section.

3.4.3 FIFO Dimensioning

The FIFO depth of the DSPIN router modifies the performance of the network. The BE throughput is highly impacted by the FIFO depth while the GS throughput is less impacted due inexistent routing congestion. Assuming that the end-to-end GS path is reserved to a unique GS traffic, a FIFO able to deliver at least 50% throughput is a good candidate for the GS FIFOs because the reserved bandwidth of GS channel is 50%. The bi-synchronous FIFO as described in Chapter 4, delivers 50% throughput with 4 words depth on the mesochronous mode and 5 words depth on the asynchronous mode. Therefore, a GS FIFO depth of 4 words is selected.

The optimal depth of the BE FIFOs depends on the application. A simulation of the network traffic can help to define an optimum tradeoff between network performance and router area. We simulated a platform of 5x5 clusters to analyze the influence of the packet length and the network charge in function of the BE FIFO depth. A traffic generator is placed on each cluster. It is possible to configure the offered load and packet size while the packet destination is randomly selected. Figure 3.21 shows the saturation threshold in function of the BE FIFO depth for different packet lengths. For BE FIFO depth between 4 and 10 words, the saturation threshold is correlated to the BE FIFO depth.

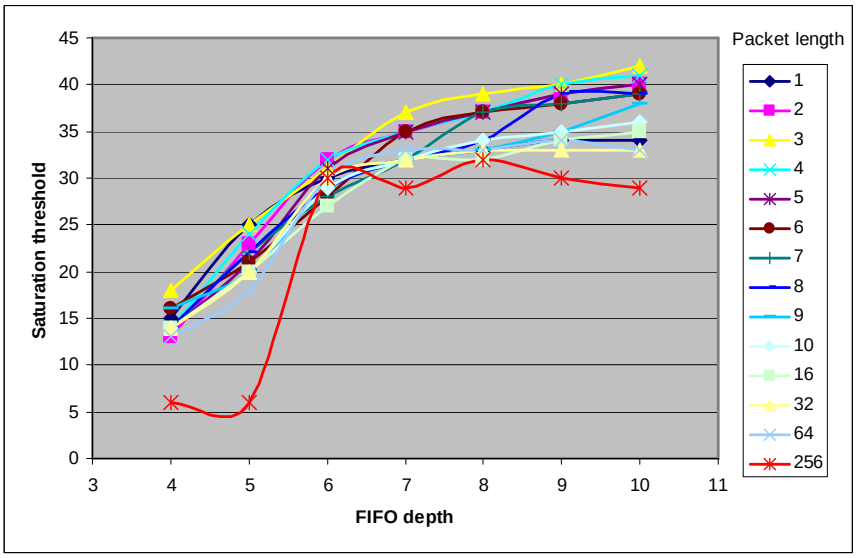


Figure 3.21 Saturation threshold in function of BE FIFO depth

Figure 3.22 shows the saturation threshold in function of the BE FIFO depth for deep FIFOs. The correlation of the BE FIFO depth with the saturation threshold is less noticeable for FIFO depths higher than 10 words. Over that depth, the packet length becomes the limiting factor. Thus, the congestion becomes more important when the packet length increased. Furthermore, it is possible to obtain a saturation threshold near 50% when the packet length is no longer than 10 flits and the FIFO depth is at least 32 words.

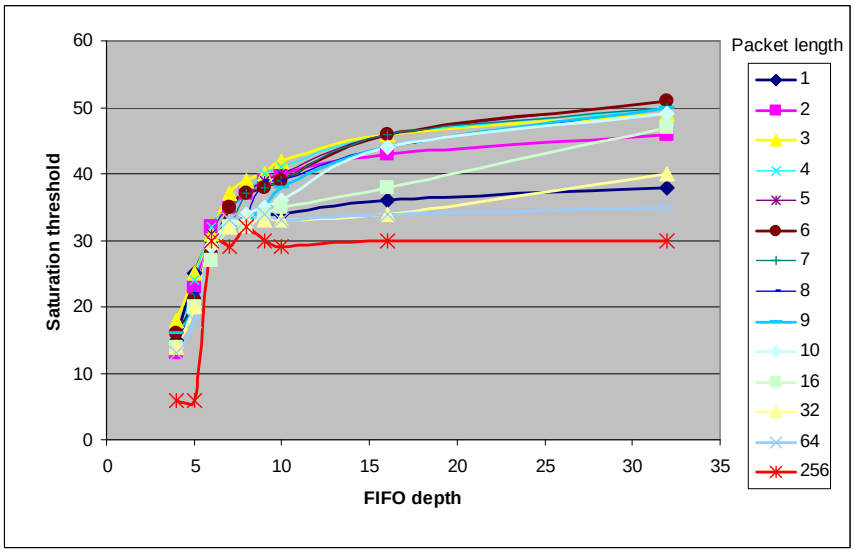


Figure 3.22 Saturation threshold in function of BE FIFO depth (up to 32 words)

Figure 3.23 shows the saturation threshold in function of the packet length. If the BE FIFO depth is higher than 5 words, a packet throughput of around 30% can be

obtained even for long packets (256 flits). FIFO depth of less than 6 words can be used only with small packet length, otherwise the saturation threshold decreases rapidly. This comes from the fact that bi-synchronous FIFO depth of 4-5 words suffer from flow-control latency penalties (see Chapter 4 for further details)

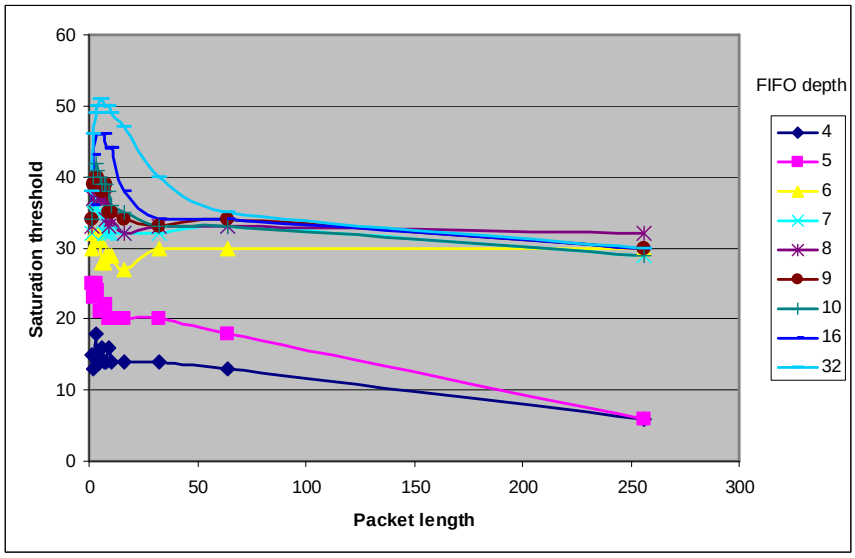


Figure 3.23 Saturation threshold in function of the packet length

Figure 3.24 shows the saturation threshold in function of the packet length between 1 and 16 words. Under these circumstances, the saturation threshold is not influenced by the packet length. The FIFO depth is the limiting factor of the saturation threshold.

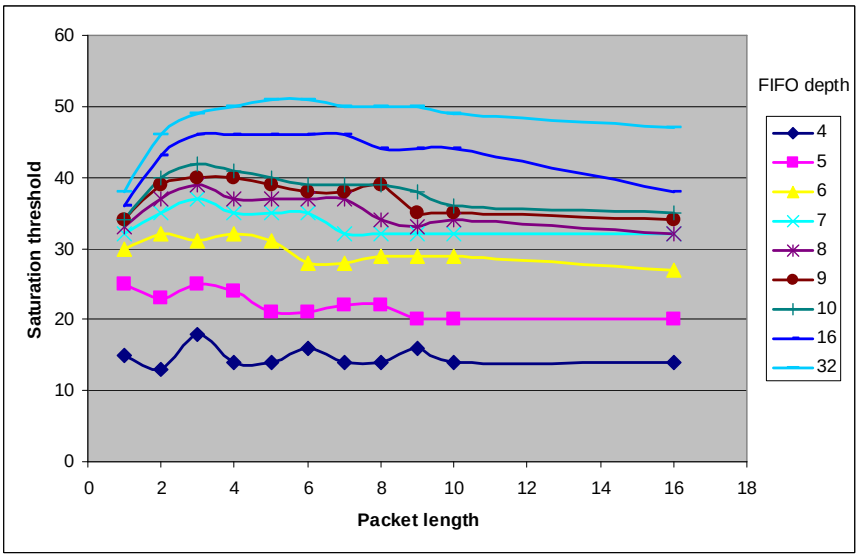


Figure 3.24 Saturation threshold in function of the packet length

The saturation threshold expresses the maximum accepted load of the network. However, the offered load on a real application is no higher than 20%. Therefore, the FIFO dimensioning can be analyzed in terms of packet latency at 20% offered load rather than on the saturation threshold. Figure 3.25 shows the mean packet latency (round trip) in function of the BE FIFO depth at 20% offered load. Under these conditions, a 7 words depth for the BE FIFO and a packet length shorter than 10 words is a good tradeoff between FIFO depth, packet latency, and router area. Hence, the mean latency for a 5x5-cluster network is 38 clock cycles. Higher BE FIFO depths do not reduce the packet latency but increases the router area. However, higher packet length increases the packet latency due to higher router congestion.

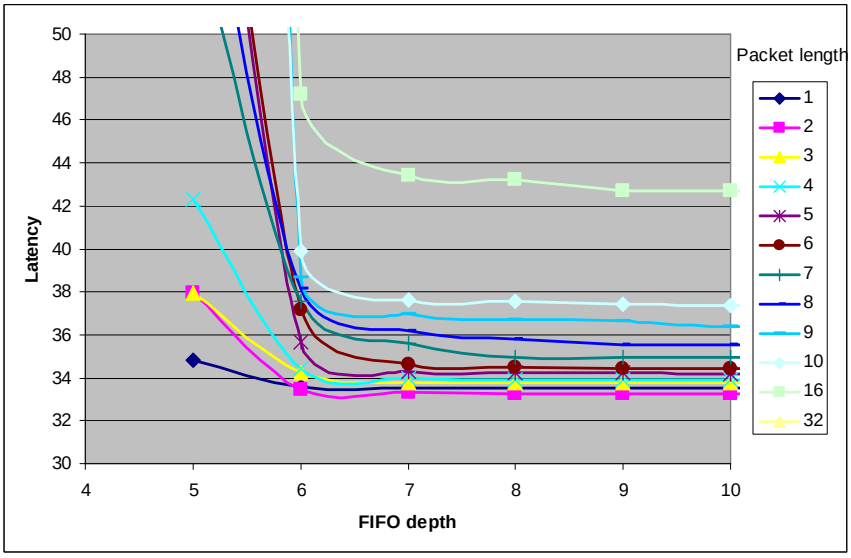


Figure 3.25 Mean packet latency in function of the FIFO depth at 20% offered load

3.4.4 Synthesis and Performance Estimation

The area evaluation of the DSPIN network was done for a 90 nm CMOS process. As all DSPIN components are synthesizable, we have computed the silicon area for the FIFOs and the routers using the STMicroelectronics GPLVT CMOS 90nm standard cell library. Moreover, the clock gating technique was used for power reduction. Table 3.2 shows the Synopsys area after synthesis of one router and the associated FIFOs: 5 BE FIFOs and 5 GS FIFOs for a clock frequency of 500 MHz. The depth of the BE and GS FIFOs are 7 and 4 flits respectively, and the flit size is 34 bits. The long wires of the router were constrained with 200ps of propagating time to simulate its physical implementation. 75% of the total DSPIN area belongs to the FIFOs; hence, the importance to optimize its performances.

Table 3.2 DSPIN area estimation (500 MHz)

Bloc	Area
5 BE FIFOs of 7x34 bits	0.0270 mm²
5 GS FIFOs of 4x34 bits	0.0156 mm²
Router without FIFOs	0.0147 mm²
Total router area	**0.0573 mm²**

As a mater of comparison between a DSPIN router with GS and a DSPIN router without GS, the area of a DSPIN router without GS is 0.040 mm². Therefore, the overhead of the GS in the DSPIN architecture is +43%, while now DSPIN has doubled the number of channels.

The router was synthesized for several clock frequencies to obtain a tradeoff between performance and area cost. Table 3.3 shows the DSPIN router area in function of the synthesized clock frequency. It is possible to synthesize the router to 833 MHz; however, its area increases by 37% over the lowest area (at 500 MHz).

Table 3.3 DSPIN router area in function of the clock frequency

Operating frequency	DSPIN area
500 MHz	0.057 mm²
666 MHz	0.067 mm²
833 MHz	0.078 mm²

3.5 Conclusion

The main objective of this chapter was the definition and implementation of a guaranteed service mechanism in the DSPIN architecture. Two different techniques have been studied, a statistical approach and the virtual channel. The statistical approach improved the latency of high priority packet by modifying the allocation priority of the router. However, the performances of this algorithm were not satisfying because the latency of the priority packet could not be hard bounded, as the latency of the high priority packets was highly influenced by the low priority network traffic. These results pushed us to implement the guaranteed service traffic using the virtual channel approach. This technique offers a good tradeoff because the most costly network resources are shared between the virtual channels.

We demonstrated that the virtual channel approach (generally used to multiplex several logical channels on the physical link between routers), can be applied to the

router itself. Two traffic classes are used, guaranteed service and best effort. The latency and the throughput of the GS traffic can be strictly bounded.

The DSPIN NoC is suited to the GALS approach. The DSPIN routers use a mesochronous clocking approach to distribute the same clock frequency to all the routers while each cluster can have its own clock frequency. This mesochronous clocking distribution allows the network to be predictable and reduces the power consumption (compared to a fully synchronous clock distribution).

DSPIN architecture is synthesizable with standard cells only, without either asynchronous or custom cells. The router itself is distributed in five modules placed on the borders of the cluster. The long wires of the DSPIN router are the intra-cluster wires while the wires interconnecting two routers are short. Thus, the timing closure is simplified, since the long wires are confined inside the isochronous island, the cluster.

A simulation platform demonstrated the efficiency of the guaranteed service communications. It has also been used to analyze the saturation threshold of the network. Moreover, the influence of the FIFO depth and the packet length has been characterized in order to maximize the network performances and reduce the latency. We showed that a BE FIFO depths of 7 words and packet length not longer that 10 words is a good tradeoff between packet latency, packet payload, and implementation cost.

Finally, the DSPIN router architecture has been synthesized on CMOS 90nm process. The silicon area is 0.057mm² at 500MHz clock frequency with 7 and 4 words per BE and GS FIFO respectively. The area overhead of the guaranteed service is 43%, while now DSPIN has doubled the number of channels.

Chapter 4

Synchronization

This chapter describes the bi-synchronous FIFO, which is capable to interface synchronous systems working with different clock signals (frequency and/or phase). Its interfaces are synchronous and its architecture is scalable and synthesizable in synchronous standard cells. The metastability situations and its latency are analyzed. Its throughput, maximum frequency, and area are evaluated in function of the FIFO depth.

The bi-synchronous FIFO uses a new encoding algorithm to simplify the synchronization of the write and read pointer. This algorithm is first detailed to introduce later the bi-synchronous FIFO architecture. The chapter summarizes with a comparison of this work with state-of-the-art architectures.

4.1 Bubble Encoding

In this section, a novel-encoding algorithm based on a token ring is demonstrated to be useful on the synchronization of pointers between two independent clock domains.

4.1.1 Token Ring

A token ring is a succession of nodes interconnected in a circular manner that contain tokens. It can be described with N registers (with enable signal) interconnected as a cyclic shift-register. Figure 4.1 shows an example of a token ring with 5 registers.

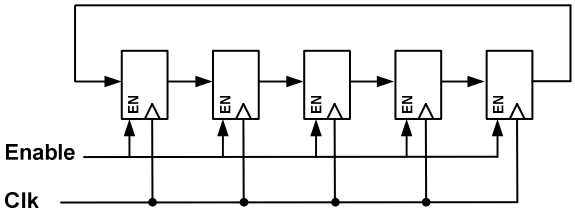


Figure 4.1 Token ring

If the enable signal is true, the content of the register is shifted (register *i* is shifted to register *i*+1, an register N-1 to register 0) at the rising edge of the clock, otherwise the register maintains the data. A token is represented by the logic state 1 of the register.

A token ring with one token can be seen as a state-machine with N states. The position of the token defines the state of the state-machine. It is also possible to define a state-machine with N states when the token ring contains two consecutives tokens, the state of the state-machine can be defined, for example, as the position of the first one.

4.1.2 Synchronizing the Token

Since the position of the tokens defines the state of the state-machine, it can be synchronized to interface two clock domains. To synchronize the state of the state-machine, a parallel synchronizer (two registers per bit) can be used, as shown in Figure 4.2.

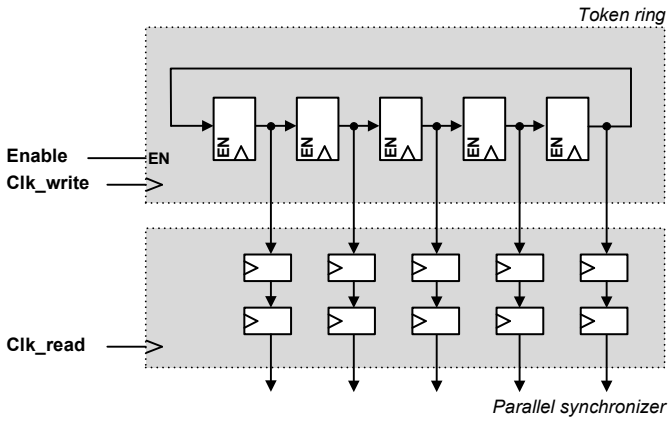


Figure 4.2 Synchronization of a token ring

However, as described in Chapter 2, the parallel synchronizer does not guarantee the correctness of the result in the case of a single token. The Figure 4.3 shows an example of synchronization. Solution A, B, C and D are all the possible solutions

when the metastability changes the register content. Solution A and B are correct synchronized since the token is well determined. Solution C is exploitable using some logic but Solution D is useless due to absence of information. In this later case, the position cannot be correctly extracted and the consumer side should wait a full clock cycle to attempt to obtain a useful data.

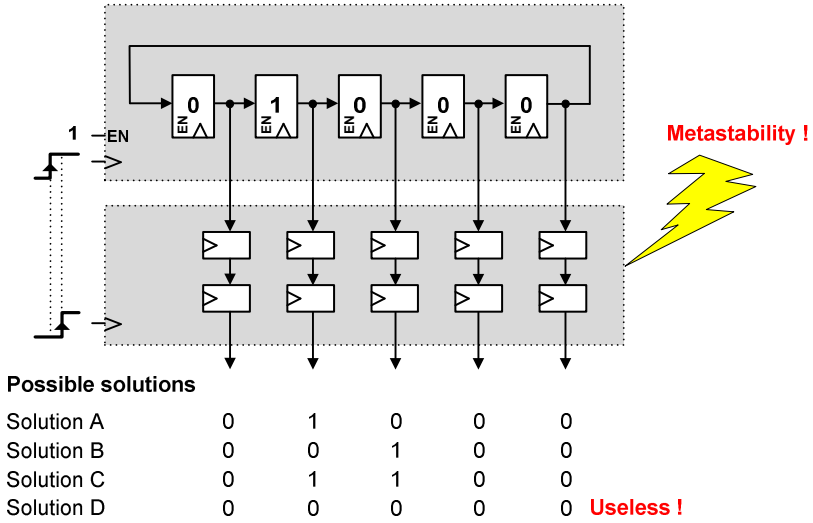


Figure 4.3 Possible solution in the synchronization of a token ring containing one token

To solve this issue, we propose to use two consecutive tokens (bubble encoding) in the token ring. As the metastability affects the changing registers, the use of two consecutive tokens prevents some registers from changing. Assuming that registers i and i+1 have the tokens, if the token ring shifts, register i+2 gets a token, register i loses its token, and register i+1 does not change (it shifts its token and gets a token). In terms of logic value, register i and i+2 change its logic state but register i+1 remains unchanged. The Figure 4.4 shows an example of synchronization. For example, we can define the position of the detected token by the position of the first logic 1 after a logic 0 (starting from the left). In this case, all solutions A, B, C, and D are correct because the token can be well defined; it is always possible to detect a transition between 0 and 1. This encoding algorithm does not avoid the metastability on the synchronizer. It just guarantees that the position of the token will be detected under any possible circumstance.

The token ring and the bubble encoding presented in this section are used on the definition of the state-machines of the bi-synchronous FIFO and will be detailed in next section.

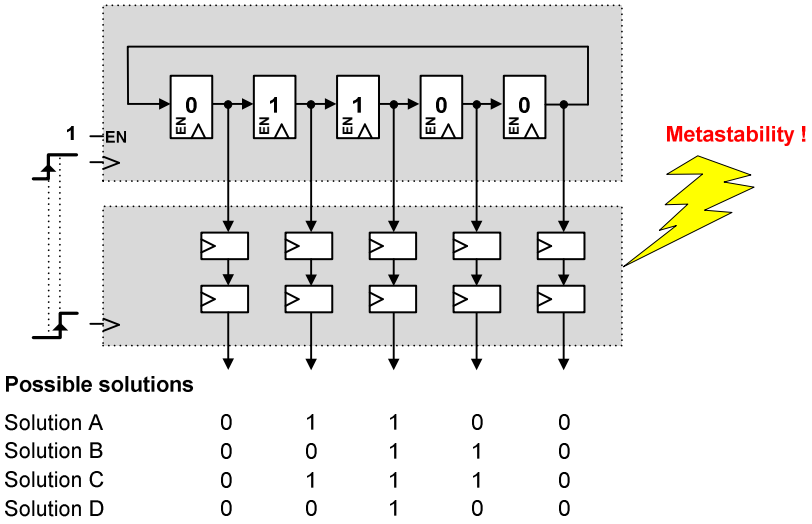


Figure 4.4 Possible solution in the synchronization of a token ring containing two successive tokens

4.2 Bi-Synchronous FIFO

This section presents the architecture of the bi-synchronous FIFO [Miro07b][Miro08]. The goal of this FIFO is to interface two synchronous systems having different clock signals (frequency and/or phase). The challenge of this architecture is to hide all synchronization issues while respecting the FIFO protocol on each interface. Furthermore, this architecture must be scalable and synthesizable in a synchronous standard-flow without using custom cells.

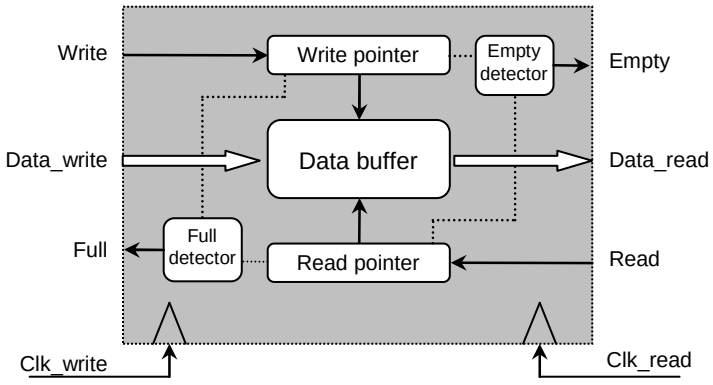


Figure 4.5 Bi-Synchronous FIFO architecture

As shown in

Figure 4.5, five modules compose the bi-synchronous FIFO architecture: Write pointer, Read pointer, Data buffer, Full detector, and Empty detector. The Write and Read pointers indicate the position to be written and to be read in the Data buffer, the

Data buffer contains the buffered data of the FIFO, and the Full and Empty detectors signal the fullness and the emptiness of the FIFO.

To better understand the bi-synchronous FIFO, its interfaces and protocol are detailed.

4.2.1 Bi-Synchronous FIFO Interface and Protocol

The bi-synchronous FIFO has a sender and a receiver interface. As shown in Table 4.1, each interface has its own clock signal, *Clk_write* for the sender and *Clk_read* for the receiver.

The FIFO protocol is synchronous; all input and output signals in the sender and receiver interfaces are synchronous to their clock signal *Clk_write* and *Clk_read*, respectively.

Table 4.1 Sender and receiver interface signals

	Signal	Description
Sender interface	Data_write	Data to be written into the FIFO
	Write	Input signal requesting a write into the FIFO
	Full	Output signal indicating the fullness of the FIFO
	Clk_write	Sender clock signal
Receiver interface	Data_read	Output data from the FIFO
	Read	Input signal requesting a read in the FIFO
	Empty	Output signal indicating the emptiness of the FIFO
	Clk_read	Receiver clock signal

The queuing and dequeuing of data elements in the FIFO follows a fully synchronous protocol. The Data_write is queued into the FIFO, if and only if, the Write signal is true and the Full signal is false at the rising edge of *Clk_write*. Symmetrically, data is dequeued to Data_read, if and only if, the Read signal is true and the Empty signal is false at the rising edge of *Clk_read*.

The clear partitioning of the sender and receiver interfaces into synchronous and independent interfaces simplifies the timing constrains analysis for all the modules connected to the FIFO ports.

4.2.2 Write and Read Pointers

The Write and Read pointers are implemented using the described token rings with the bubble-encoding algorithm. The position of the tokens determines the position of the pointer. The position of the *Write_pointer* is defined by the position of the register containing the first token (starting from the left) as shown in Figure 4.6a.

Likewise, the position of the *Read_pointer* is defined by the position of the register after the second token (starting from the left). The Full and Empty detectors exploit this particular definition of the pointers and will be explained hereinafter.

The *Write_pointer* shifts right when the FIFO is not full and the Write signal is true. Likewise, the *Read_pointer* shifts right when the FIFO is not empty and the Read signal is true.

As the write and read interfaces belong to different clock domains, the token rings are clocked by their clock signal, *Clk_write* and *Clk_read* respectively.

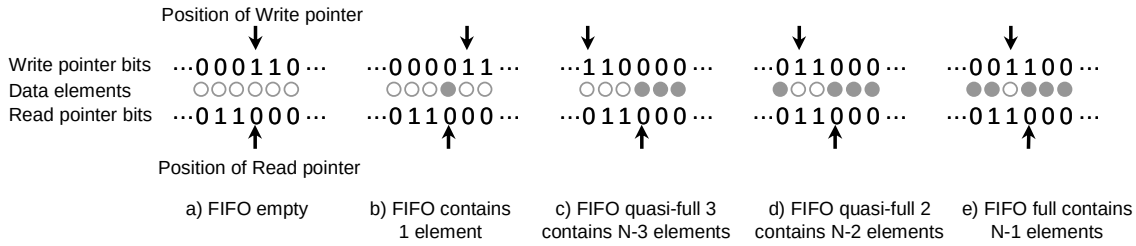


Figure 4.6 Write and Read pointer position definition and Full and Empty conditions in terms of tokens position

4.2.3 Data Buffer

The Data buffer module is the storage unit of the FIFO. Its interfaces are: *Data_write*, *Data_read*, *Write_pointer*, *Read_pointer*, and *Clk_write*. It is composed of a collection of data-registers, AND gates, and tri-state buffers as shown in Figure 4.7.

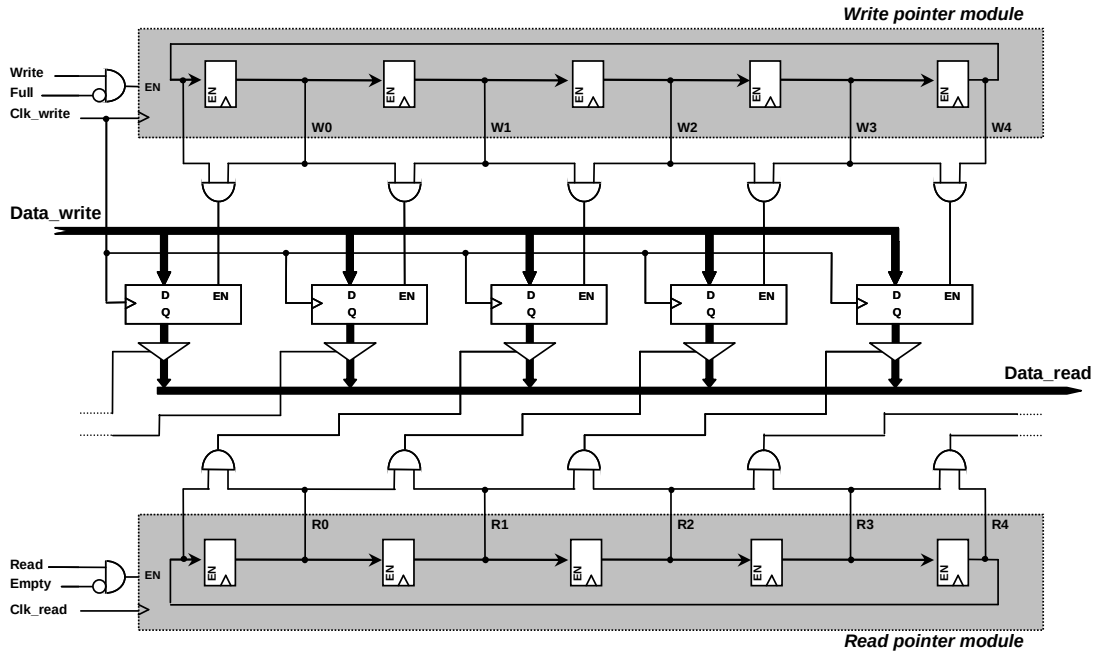


Figure 4.7 Write pointer, Read pointer, and Data buffer detail

The input data, *Data_write*, is stored into the data-register pointed by the *Write_pointer* at the rising edge of *Clk_write*. AND gates recode the *Write_pointer* into a one-hot encoding which controls the enable signals of the data-registers. Likewise, the *Read_pointer* is recoded into one-hot encoding which controls the tri-state buffer on each data-register. Finally, the *Data_read* signal collects the outputs of the tri-state buffers. It is also possible to replace the tri-state buffers with multiplexers to simplify the Design for Test (DfT) of the FIFO.

The width and number of data-registers determine the width and the depth of the FIFO. The depth also determines the range of the Write and Read pointers.

4.2.4 Full Detector

The Full detector computes the *Full* signal using the *Write_pointer* and *Read_pointer* contents. No status register is used as in the J. Jex et al. [Jex97] or Chelcea-Nowick [Chelcea04] solutions. The Full detector requires N two-input AND gates, one N-input OR gate, and one synchronizer, where N is the FIFO depth (Figure 4.8).

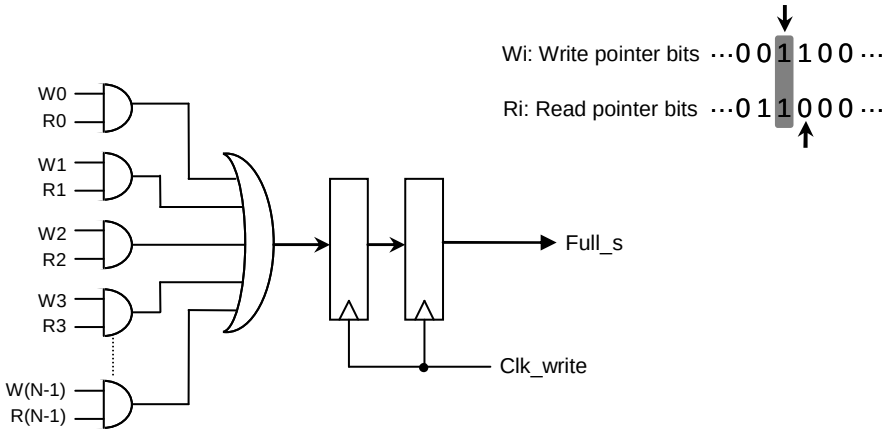


Figure 4.8 Full detector detail

The detector computes the logic AND operation between the *Write* and *Read* pointer bits and then collects it with an OR gate, obtaining logic value 1 if the FIFO is Full or quasi-Full (cases c, d, e in Figure 4.6). If $\exists i, Wi = Ri = 1$ then the output of the OR gate is asserted, meaning that the FIFO is going to be full. The obtained value is finally synchronized to the *Clk_write* clock domain into *Full_s* signal.

Since the synchronization has a latency of one clock cycle and the synchronization of the OR output signal can potentially be metastable, the detector has to anticipate the detection of the Full condition. For this reason, the output of the OR gate detects the Full and the two quasi-Full conditions.

The Full detector in Figure 4.8 could be optimized since the synchronization latency inhibits, in some cases, the FIFO from being completely filled. For example, if the FIFO is in the situation of Figure 4.6c, and the sender does not write any other data, the *Full_s* will be asserted even if the FIFO is not filled completely.

An improved Full detector implementation would be more complex (as the Empty detector), and would therefore require greater chip area. However, a non-optimal Full detector does not penalize the throughput of the FIFO as much as a non-optimal Empty detector. For example, assuming an optimal Empty detector and a non-optimal Full detector, the Full condition occurs when the receiver is not able to consume all the data. In this case, even with a non-optimal Full detector, the receiver limits the throughput of the FIFO. Therefore, design effort and chip area should be devoted to improving the performance of the Empty detector.

Even when using a non-optimized Full detector, a low cost Full detector optimization can improve its performance. Figure 4.9 shows an additional module connected to the *Full_s* signal, which improves the Full detector. The module's operation is as follows: if the writer was not writing before asserting the *Full_s* signal, the *Full* signal is delayed one clock cycle, giving a second chance to the writer to fill completely the FIFO.

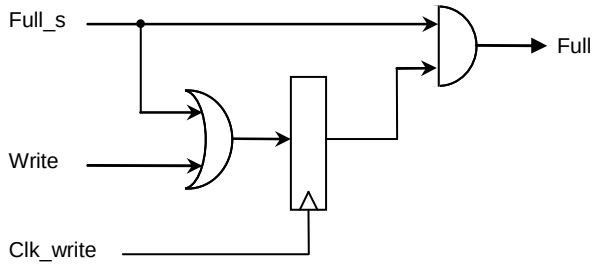


Figure 4.9 Full detector optimizer

4.2.5 Empty Detector

The implementation of the Empty detector is similar to the Full detector because both use the Write and Read pointers. As seen in the previous paragraph, the Full detector has to anticipate the detection of the Full condition to avoid FIFO overflow. As the Empty detector performance is correlated to the FIFO throughput, its detection has to be optimized, and no anticipation detector should be used.

Figure 4.10 shows the Empty detector for a five word FIFO. Firstly, the *Write_pointer* is synchronized with the read clock into the *Synchronized_Write_pointer* (SW) using a parallel synchronizer. Next, the *Read_pointer* is recoded into the

AND_Read_pointer (AR) using two-input AND gates, reused from the Data_buffer module. The output of AR is a one-hot encoded version of the *Read_pointer*. Finally, the Empty condition is detected comparing the SW and AR values using three-input AND gates. As the metastability can perturb some bits of the SW (as seen on Figure 4.4), each pair of consecutive bits is compared to find a transition between 0 and 1. Their analysis is as follows, if the values of SWi = 0 and SWi+1 = 1 that means that the SW pointer is on position i+1. Furthermore, when ARi = 1 that means that the AR pointer is on position i+1 (Figure 4.10).

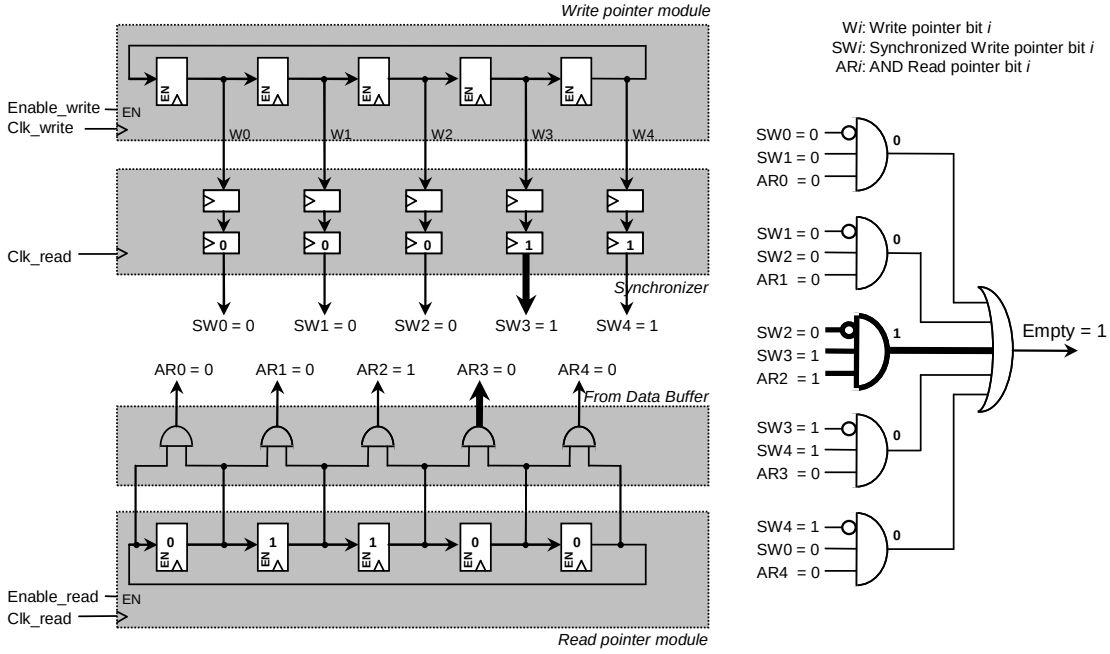


Figure 4.10 Empty detector detail

The FIFO is considered empty (see Figure 4.6a) when the *Write_pointer* points the same position of the *Read_pointer*. This can be detected when SWi=0, SWi+1=1 and ARi=1 for any i. These comparisons are computed by means of the three-input AND gates. Finally, a N-input OR gate collects all the values of the three-input AND gates to generate the Empty signal. This N-input OR gate and the one on the Full detector can be decomposed with $\log_2 N$ levels of two-input OR gates.

The latency introduced by the synchronization of the *Write_pointer* cannot corrupt the FIFO, because a change in this pointer cannot underflow/overflow the FIFO, it just introduces latency into the detector.

The advantage of the bubble-encoding algorithm in this detector relies on the guaranteed detection of the *Write_pointer* position.

4.2.6 Mesochronous Adaptation

The FIFO architecture was originally designed to interface two fully independent clock domains. However, it can be adapted to interface mesochronous clock domains where the sender and the receiver have the same clock frequency but different phase. The difference of phase can be constant or slowly varying and this predictability can be used to avoid the metastability situations [Mu01] [Mesga04].

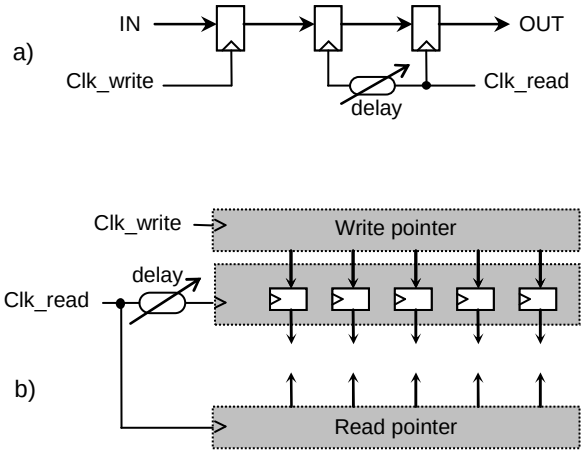


Figure 4.11 Mesochronous adaptation

The proposed adaptation lowers the FIFO latency by reducing the number of registers on the synchronizer module. The two rows of registers on the synchronizer can be reduced to a single row of registers as shown in Figure 4.11b. The remaining row of registers is clocked using a delayed version of the read clock. This delay must be chosen to exchange the data without metastable situations (Figure 4.11a). The delay can be a programmable delay, or any other metastability-free solution, as for example the Chakraborty-Greenstreet [Chakra03] architecture allowing the FIFO to work also on plesiochronous (small difference of frequency) clocks. Likewise, if the write and read clock are out of phase by 180º (clock-inverter), no programmable delay is needed because, by-construction, the communication is free of metastability. Figure 4.12 shows this construction where a clock inverter is added and the clock signals are delayed by an unknown delay. Under these circumstances and assuming that the registers are very close (zero wire delay), the communication is free of metastability, if only if, the *setup_time*, *hold_time*, and *access_time* are respected. Thus, the interface is free of metastability also if the difference of phase varies under the metastability free window.

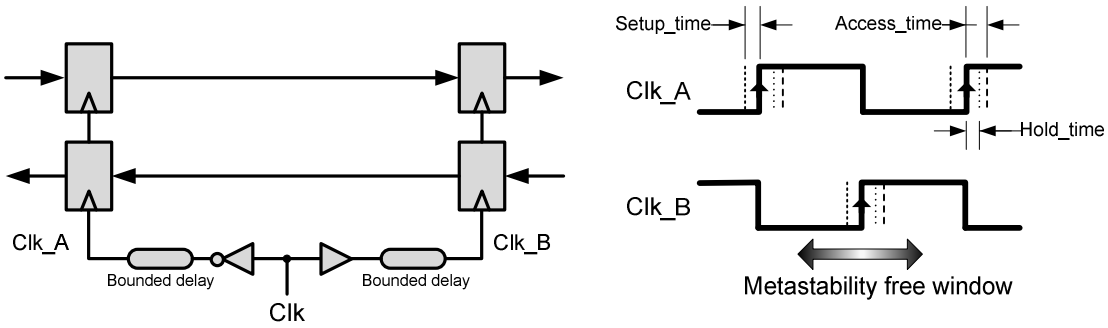


Figure 4.12 Metastability free window with inverted clock signals

This mesochronous adaptation of the bi-synchronous FIFO is simple and allows switching between mesochronous and asynchronous modes. This adaptation is interesting in the design of a multi-million gate SoC in deep sub-micron technology, where the delay of long wires can drastically vary with temperature, voltage, and process. In such a system, the mesochronous clock distribution could fluctuate to an undesirable metastable situation, making the FIFO data useless. By switching the bi-synchronous FIFO into the asynchronous mode, robustness against metastability is improved, preventing the SoC from requiring redesign.

4.3 Simulation and Analysis

Both synthesizable VHDL models and cycle accurate SystemC models of the bi-synchronous FIFO have been designed. We have simulated the bi-synchronous FIFO to characterize its latency, throughput, frequency, and area.

4.3.1 Latency Analysis

As the sender and the receiver have different clock signals, the latency of the FIFO depends on the relation between these two signals.

The latency of the FIFO can be decomposed in two parts: the state machine latency and the synchronization latency. As the state-machines are designed using Moore automates, its latency is one clock cycle. Two registers compose the synchronizers and its latency is ΔT plus one clock cycle. Where ΔT is the difference, in time, between the rising edges of sender and receiver clock. As this difference is between zero and one Clk_read clock cycle, the latency of the bi-synchronous FIFO is between two and three Clk_read clock cycles. Figure 4.13 shows the detail of the latency. Sync_1 and Sync_2 are the synchronization registers. The latency of the bi-synchronous FIFO is equivalent to the latency of the J. Jex et al. [Jex97] solution. This

latency can be lower, but the robustness to the metastability would be penalized [Dike99] [Ginosar03].

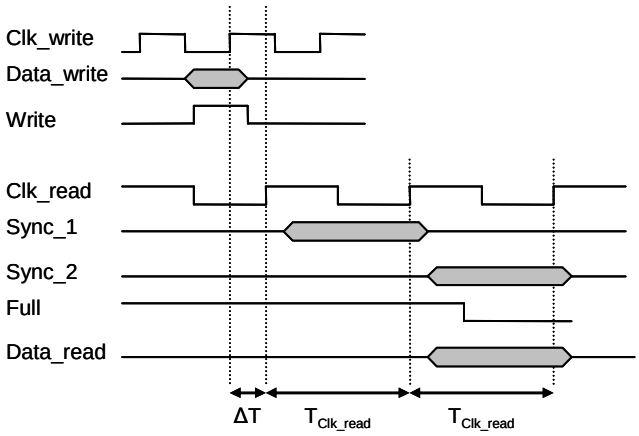


Figure 4.13 Latency analysis

When the bi-synchronous FIFO is adapted to a mesochronous clock distribution, the latency of the FIFO is reduced, because a single register replaces the two-register synchronizer. In addition, the ΔT is constant as the difference of phase is constant. In that case, the latency of the FIFO is one clock cycle plus ΔT, as shown in Figure 4.14.

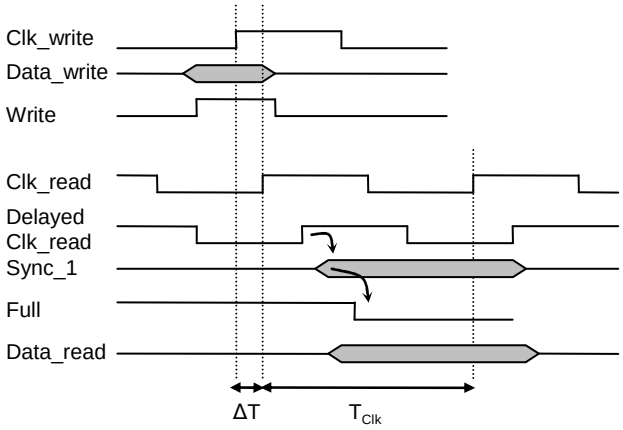


Figure 4.14 Latency analysis with mesochronous adaptation

4.3.2 Throughput Analysis

The throughput of the bi-synchronous FIFO was analyzed as a function of the FIFO depth. As the synchronizers add latency, the flow control is impacted by the FIFO depth. In case of deep FIFO, the synchronizers do not decrease the FIFO throughput since the buffered data compensate the latency of the flow control. Table 4.2 shows the minimum FIFO depth for 50% and 100% throughput for the asynchronous and mesochronous. For FIFO depth of 6 or above, the synchronization

latency has no influence on the flow control and the FIFO is able to deliver one word per cycle (100% throughput) even on asynchronous clock relation. For the asynchronous analysis, the write and read clock signals frequencies are similar, otherwise it is not possible to obtain 100% throughput.

Table 4.2 Minimum FIFO depth in function of the clock relation and required throughput

	Minimum depth for 50 % throughput	Minimum depth for 100 % throughput
Asynchronous	5	6
Mesochronous	4	5

4.3.3 Area and Frequency Estimation

The area and frequency estimation of the FIFO was computed once synthesized on CMOS 90nm GPLVT STMicroelectronics standard cells. Different FIFO depths are used to illustrate the scalability of the architecture and its performances in terms of maximum frequency. To minimize the power consumption, a clock gating technique is used. Two architectures were synthesized, one with the tri-state buffers and another with multiplexers.

Table 4.3 shows the area and frequency estimation of a 32-bit bi-synchronous FIFO in function of the FIFO depth. Note that the maximum frequency of the write clock is greater than the one of the read clock. The limitation of the read clock is due to the Empty detector.

The architecture with tri-state buffers has greater area than the one with multiplexers, while the maximum clock frequency of the read part with tri-states is greater than the one with multiplexers, since the multiplexers are decoded in a $\log_2 N$ manner rather than in parallel.

Table 4.3 Area and frequency in function of FIFO depth

Type	FIFO Depth	Area (μm^2)	Max. Write Freq. (MHz)	Max. Read Freq. (MHz)
Mux	4	3304	2000	1110
	8	6581	2000	1000
	16	13384	2000	769
Tri-state	4	4082	2000	1428
	8	8032	2000	1250
	16	16101	2000	1110

4.3.4 Comparison with other Existing Designs

This architecture has been compared with similar architectures to analyze its area and latency. As its architecture is synthesizable with standard cells, the comparison with the others is performed with synthesizable architectures.

The selected architectures are a register-based Gray FIFO and the J. Jex et al. [Jex97] FIFO. The register-based Gray FIFO uses the Gray-code to implement the Full and Empty detectors. The write and read pointers are coded on natural binary code. The pointers are converted to Gray-code, synchronized, reconverted to natural binary code, and finally compared to compute the Full and Empty signals. On the other hand, the J. Jex et al. FIFO uses one-hot coding algorithm for the write and read pointer. A status register computes the filled cells of the FIFO. The status register bits are set by the write side and reset by the read side. A parallel synchronizer and a combinational logic compute the Empty signal. Likewise, a combinational logic and a two-flop synchronizer compute the Full signal. Appendix A contains detailed information of both architectures.

The three architectures where modeled using VHDL RTL. Optimized implementations of the FIFOs were performed for 4, 8 and 16 words depth. The word size was fixed to 32 bits. All the architectures were synthesized using the same CMOS 90nm GPLVT STMicroelectronics standard cells library and using the same timing constraints file. A clock-gating technique was applied but no tri-state buffers were used. Table 4.4 shows the estimated area and the area overhead percentage of these architectures compared to the proposed solution.

Table 4.4 Area and overhead comparison between other existing designs

FIFO Depth	This Design μm^2	Register-based Gray FIFO μm^2 (%)	J. Jex et al. [Jex97] μm^2 (%)
4	3304	5113 (+54%)	3364 (+1.8%)
8	6581	9702 (+47%)	6858 (+4.2%)
16	13384	20364 (+52%)	14362 (+7.3%)

The register-based Gray FIFO has a 50% bigger area than the proposed architecture. Even if the number of synchronizers is lower than our architecture, the Gray code algorithm adds complexity to the Full and Empty detectors.

The J. Jex et al. [Jex97] architecture has similar complexity as ours, but its area increases more than ours when the FIFO depth increases. Moreover, its Full detector

is not optimized and suffers the same problem of the non-optimized Full detector presented in Figure 4.8. Furthermore, the J. Jex et al. FIFO requires a FIFO depth of at least 14 words to archive 100% throughput while ours requires just 6.

In terms of FIFO latency, all three have the same latency, 2-3 clock cycles, since all of them use Moore state-machines and two flip-flops synchronizers.

4.4 Conclusion

A new bi-synchronous FIFO architecture has been implemented and analyzed. It is well suited to interface different systems working with independent frequency and/or phase clock signals. It uses a novel encoding algorithm combined with an astute definition of the FIFO pointers that avoids the utilization of status registers. The write and read pointers are directly combined to obtain the Full and Empty signals.

Both read and write interfaces are fully synchronous. Moreover, its architecture is synthesized using a synchronous standard cell design flow. None of its modules requires custom cells.

A simple mesochronous adaptation is proposed which reduces the latency of the FIFO. Its latency is 2-3 clock cycles in asynchronous mode, and 1-2 clock cycles in mesochronous mode.

The FIFO throughput depends on the FIFO depth. Throughput is 100% when the FIFO depth is six or above.

Using CMOS 90nm GPLVT STMicroelectronics standard cells, we have synthesized and analyzed the FIFO area and maximum frequency for different FIFO depths. Two architectures are analyzed, one with tri-state buffers and another with multiplexers. A 32-bit bi-synchronous FIFO with eight words depth requires 6581μm^2 and its maximum clock frequency is 1GHz.

The comparison with previous synthesizable asynchronous FIFOs shows a better integration density for the same data latency.

The bubble encoding and the architecture of bi-synchronous FIFO have been patented by STMicroelectronics [Miro07b][Miro08].

Chapter 5

DSPIN Physical Implementation

In this chapter, we present a physical implementation of the DSPIN architecture on the stream-oriented FAUST platform developed by CEA-Léti. The details of this platform can be found in Chapter 2. The network-on-chip of FAUST (ANOC) is replaced by the DSPIN NoC. The main goal of this experiment is to prove that the DSPIN architecture can be easily integrated in an industrial design flow using commercial tools for physical synthesis. The details of this migration are in Appendix B.

Section 5.1 describes the Front-End implementation of the DSPIN network-on-chip in the Faust platform. The Back-End implementation including detailed floorplanning and clock distribution is described in Section 5.2. In Section 5.3, the implementation is validated with back-annotation simulations and the DSPIN performances are extracted. Finally, in Section 5.4, a comparison between ANOC and DSPIN designs in a 130nm technology is carried out in terms of area, throughput, packet latency, power consumption, and programmability.

5.1 Front-End Implementation

DSPIN architecture have been designed to be synthesizable on standard cells and easily implemented on a synchronous digital flow. Moreover, its architecture is optimized in terms of critical path and power consumption.

5.1.1 DSPIN Critical Paths Analysis

The critical path of DSPIN is designed to maximize the clock frequency without having to pipeline the long wires. The flits of the packets are only stored on the input

FIFOs; there is no register between two FIFOs. Moreover, there are no register-to-register paths doing a round trip over the long wires. Thereby, the influence of the long wires delay is minimized to just one-way path. Figure 5.1 shows the wire paths between the west input FIFOs to the FIFOs of the east neighbor router.

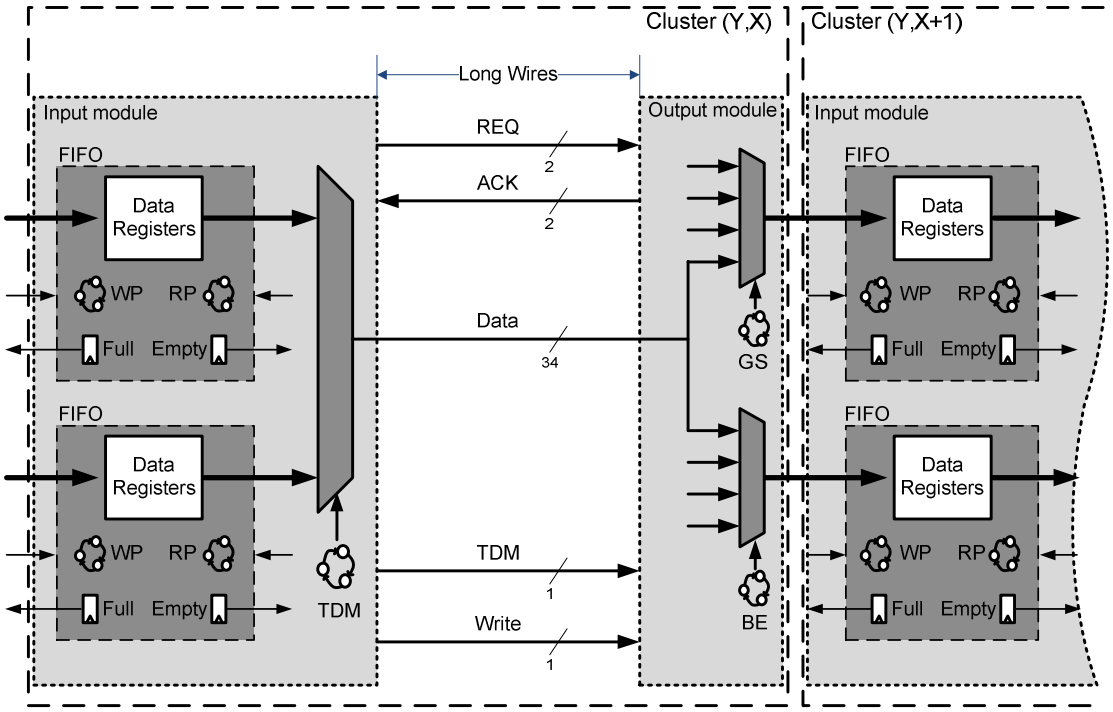


Figure 5.1 Paths between west input FIFOs to FIFOs on the east neighbor router

The analysis of the critical path after synthesis shows that the critical path starts on the detection of the Empty condition on the west FIFO, crosses the long wires through the wire *Write* arriving on the East module, passes through a multiplexer, arrives to one of the FIFOs, and controls the *Write-pointer* (WP) of this FIFO.

5.1.2 GALS Implementation

The DSPIN router-to-router links are mesochronous as a GALS implementation is used. Towards that end, the clock signals of neighbor routers have to be inverted for the correct operation of the mesochronous FIFO. With the clock signal distribution showed on Figure 5.2, neighbor routers have inverted clock phase. Therefore, the routers placed on the black boxes of Figure 5.2 have a clock inverter while those placed on white ones have a clock buffer. These cells have to be preserved during the synthesis of the circuit; otherwise, the tool eliminates them. Hence, a *set_dont_touch_network* statement is used on the clock signals.

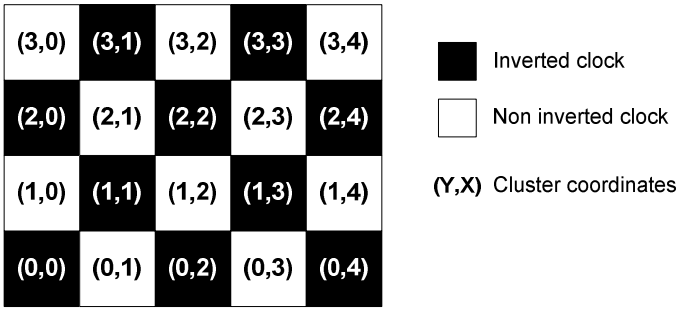


Figure 5.2 DSPIN clock phase for the FAUST implementation

For the asynchronous interfaces like router-to-subsystem, the bi-synchronous FIFO interfaces, by construction, the asynchronous interfaces. A *set_false_path* statement has to be declared between the two clocks domains.

5.1.3 Clock Gating

A first physical implementation of DSPIN without clock gating helped to validate the back-end flow. This implementation was successfully accomplished but the power consumption was not satisfying. Without clock gating, 86% of the DSPIN router power is consumed by the registers. Moreover, the clock tree of a DSPIN router consumes as much power as the router itself. Consequently, the clock-gating technique was used on a second physical implementation to overcome this limitation.

With the clock gating, it is possible to save power on the router itself and on its clock trees. Moreover, the clock gating helps to reduce the area of the bi-synchronous FIFOs. The area of a 34-bit data register with a write-enable input is higher than a 34-bit data register controlled by a clock-gating cell. Consequently, the area of the DSPIN router is reduced by 10-15% when the clock gating is used.

The introduction of the clock gating technique in the DSPIN architecture has been modulated to maximize the power saved without reducing its performances. A naïve implementation of the clock gating in the DSPIN would reduce the maximum clock frequency because the enable signal of the clock-gating would pass through the long wires and would elongate the critical path. In order to avoid an elongated critical path, two methods are used. Firstly, not all the registers are gated. Secondly, the enable signal of the clock-gating cells is generated locally, thus it does not cross the long wires. The detail of the clock-gating enable signal is detailed hereinafter.

A DSPIN router contains 2630 register and 90% of them are FIFO data registers. Therefore, we have chosen to clock-gate the FIFO data registers and not all the registers of the system, hence, the power saving is maximized without modifying to much the router architecture.

In order to generate the clock-gate enable signal near the FIFO registers, the FIFO architecture is modulated and a new Wake_up signal is generated. The design of the bi-synchronous FIFO brings naturally the introduction of the clock gating on the data registers because all the registers are controlled by write-enable signals. However, some modifications have been introduced not to elongate the critical paths when it is used on the DSPIN router. The FIFO *Write* signal controls the validity of the input data and this signal is on the DSPIN critical path. Thus, a special signal called *Wake_up* is created to control the write-enable signal of the data registers without any relationship with the FIFO *Write* signal. Therefore, a FIFO data register is clocked, if only if, the *write_pointer* is pointing it and the *Wake_up* signal is asserted. The FIFO *Write* signal only modifies the *write_pointer*, it does not affect the clock-gating cell. Hence, the DSPIN critical path is not elongated and all the data registers are clock-gated.

In order to avoid crossing the DSPIN long wires to control the *Wake_up* signal, this signal is generated locally. The *Wake_up* signal is generated on the BE and GS state-machines, which are on the neighbor router module (Figure 5.3). The *Wake_up* signal is asserted when the output port, of the neighbor router, is allocated to a virtual channel. Moreover, this signal is generated near the FIFO, thus the wire delays can be neglected, and it does not affect the critical path. The *Wake_up* signal is asserted as long as the output port is allocated to a virtual channel, even when no flit is been transferred.

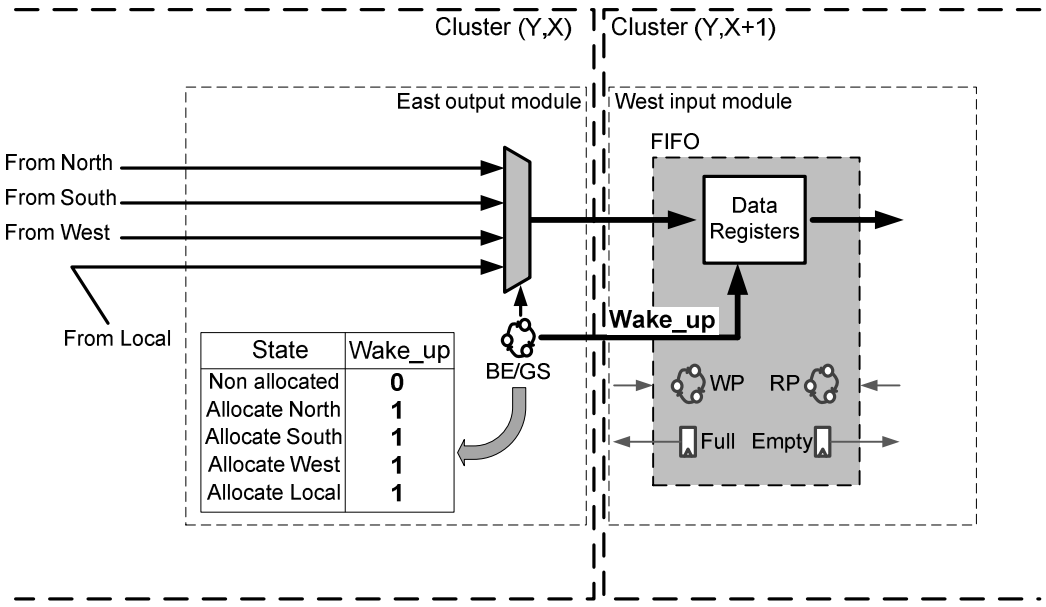


Figure 5.3 Wake_up signal definition

5.1.4 Reset Signal

As the system follows the GALS paradigm, all the long distance signals must be considered as asynchronous. The *Reset* signal is distributed along the entire circuit and properly buffered to guarantee a maximum transition time. However, this signal has to be resynchronized to guarantee a clear state after reset. Therefore, each DSPIN router has a *reset* synchronizer as the one showed in Figure 1.19.

5.1.5 Functional Validation

The application selected to validate the architecture is a SISO-MC-CDMA data-streaming application called Matrice [Berens05]. It consists in transmitting and receiving frames using OFDM and CDMA techniques, with a data rates up to 100 Mbits/s. We focused on the Matrice receiver (RX) partition, which requires 10 IP-blocks from the complete FAUST platform. For this application, the NoC interconnect support an aggregated throughput up to 10.6 Gbits/s to maintain the real-time constraints imposed by the OFDM frame rate. An OFDM frame must be processed in less than 650µs. A detailed description of the frame composition and decoding method can be found in [Berens05].

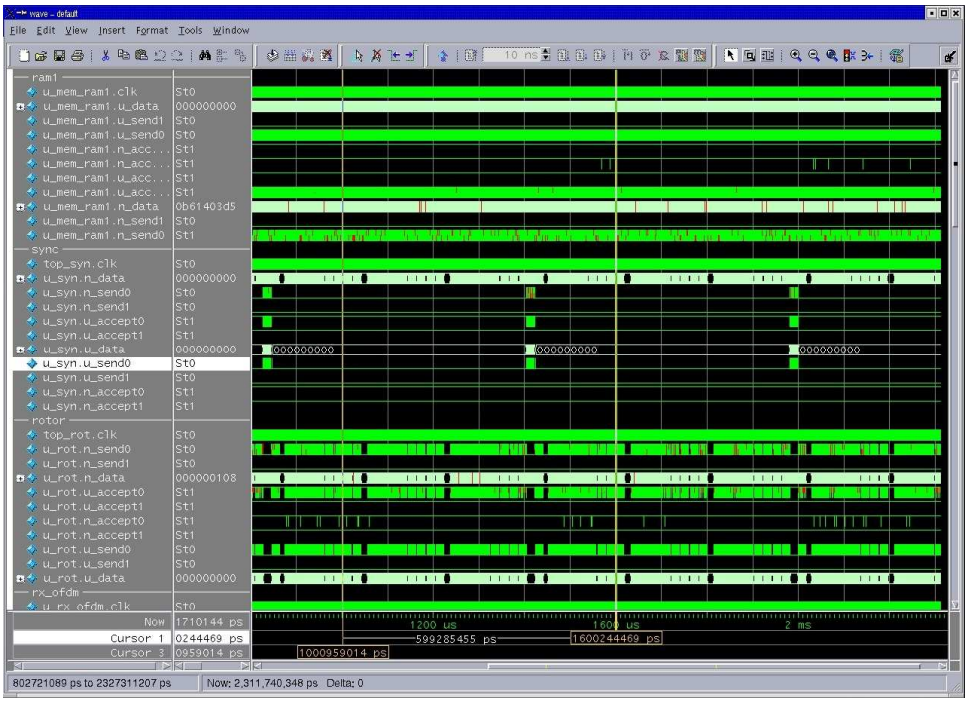


Figure 5.4 FAUST simulation

Once the DSPIN architecture is adapted for the FAUST platform (see Appendix B), a global VHDL RTL simulation is performed to verify the correctness of the architecture. This simulation is performed using real OFDM data values. Thus, the

correct demodulation of successive OFDM frames is completed and used to dimension the FIFO depth of the DSPIN routers (see Appendix B). Figure 5.4 shows the blocks RAM1, SYNC and ROTOR while demodulating two OFDM frames. The SYNC module detects the beginning of an OFDM frame and sends some packets to synchronize the OFDM demodulation module.

5.1.6 Synthesizing FAUST

Complex SoCs, as the FAUST circuit, are designed to be partitioned and synthesized as independent modules. Each module can be synthesized separately, and then finally assembled without running a RTL synthesis on the top level. The synthesis of the DSPIN routers follows the same methodology. Each DSPIN routers is synthesized separately as the router function depends on the router coordinates (Y_0,X_0).

5.1.6.1 Synthesizing the DSPIN routers

The VHDL RTL code of the DSPIN router is generic. The FIFO depths, the (Y_0,X_0) coordinates, and flit size can be modified by the template parameters. On the FAUST chip, all the DSPIN routers use the same FIFO depth and same flit size, just the (Y,X) position is modified.

The synthesis tool requires a timing-constrains file to properly optimize the design. As the routers is synthesized as an independent module, the input and output pins of the DSPIN router are properly characterized in terms of timing slack using the *set_input_delay* and *set_output_delay* statements. Moreover, the long wires delays are also considered by setting a propagation delay of 300ps over these wires. Towards that end, the *set_max_delay* statement is used.

CMOS 130nm technology with low-power cell libraries, low Vt transistors are used to synthesize the DSPIN routers. Nonetheless, the synthesis is successful up to 333MHz, which is enough for the FAUST application. The router footprint after synthesis is 0.150mm² and 0.134mm² for the non-clock-gating and the clock-gating implementation.

5.1.6.2 Synthesizing the top circuit with DSPIN

Once all the modules and routers are synthesized, the top-cell is obtained by assembling the gate-level netlist files of the modules; no RTL code is synthesized on the top. Towards that end, all the gate-level files are loaded into the synthesis tool and linked together. Moreover, the timing-constraints file of each module is also load

into the tool to generate a global timing constraints file. Finally, the correctness of the netlist is verified using a static timing analysis tool.

5.2 Back-End Implementation

In this section, the Back-End implementation of the DSPIN architecture in the FAUST platform is described. The floorplanning of the DSPIN router modules is detailed as well as the FAUST modules. In order to compare the ANOC with the DSPIN implementations, both implementations have the same chip area and use the same 130nm CMOS technology. The clock distribution network for the mesochronous DSPIN clock is built using a simple implementation method. Finally, the mesochronous and asynchronous communications are constrained and implemented, using timing constraint statements.

5.2.1 Floorplanning

Complex SoCs as the FAUST circuit use hard macro cells. These devices, (memory banks, processors…) are designed as stand-alone devices and are finally imported into the SoC. The performances of these hard macro cells are optimized, but they introduce constraints related to by their shape, their area, and the wire levels used. The floorplanning design of a SoC requires considering these constraints and adapting the rest of the circuit to meet the circuit performances.

The implementation using a GALS approach requires to physically partition the SoC into independent areas. This partitioning is done using regions. A region is a floorplanning delimiter that conditions all the cells of a module to be placed inside the defined area. However, the region does not define an exclusive area, because cells of other modules can be placed inside this area. The floorplanning using regions gives the designer the flexibility to place the DSPIN router modules on the borders of the clusters. Thus, the DSPIN routers are floorplanned taking into consideration the cluster dimension and obstructions (memory banks, processor). DSPIN routers use five regions, one for each DSPIN module (North, South, East, West, and Local). Figure 5.5 shows the FAUST floor-plan with DSPIN routers. The clusters are delimited by the big colored rectangles while the small filled rectangles are the DSPIN modules. The N, S, E, W, and L filled boxes denote the North, South, East, West, and Local DSPIN modules respectively. The DSPIN modules color is the same as the cluster color to identify the router of the cluster.

To minimize the inter-router wires, the DSPIN North, South, East, and West modules have to be placed in front of the neighbor router modules (East module is placed near the West module of its neighbor router …).

The DSPIN Local module can be placed in any place inside the cluster area. We have successfully tried various positions: on the center of the cluster and on the cluster sides. Nevertheless, a central position is not suitable due to higher wire congestion between the router and the local subsystem. In order to reduce this wire congestion, the Local modules placed on the center have larger region area, thus reducing the placement density. The DSPIN local module can be placed where the designer consider to be more efficient.

In principle, the DSPIN routers placed on the circuit SoC sides should not have useless ports. For example, the bottom-left SoC router should only have the North, East, and Local ports. For this first implementation, all the routers where synthesized with all the ports even if they do not use them. In an industrial version of the DSPIN, the routers placed on the SoC sides will not contain useless ports thus reducing the circuit area.

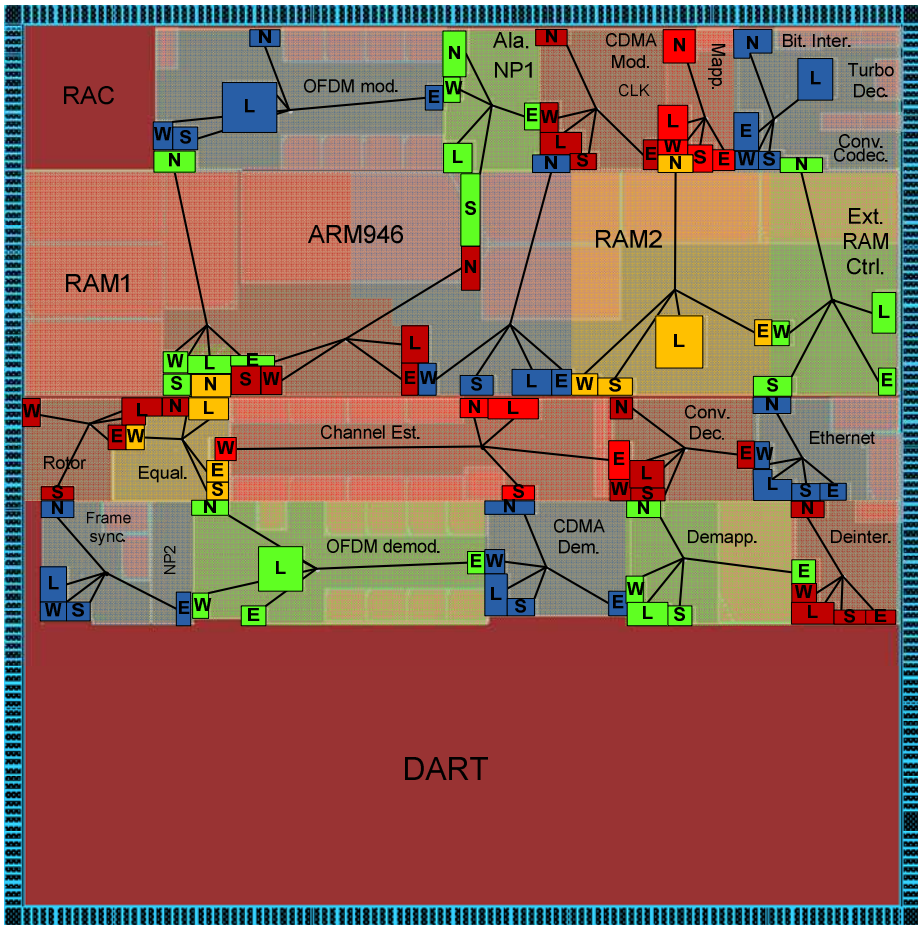


Figure 5.5 FAUST floor-plan with DSPIN

The top left red box and the bottom red box are unused regions of the circuit. On the original FAUST chip, these regions are occupied by the RAC and DART hard macro devices respectively. However, these modules are not used by the OFDM modulation/demodulation application and they are not implemented. Nevertheless, their areas are reserved to implement the FAUST circuit with DSPIN using the same area constraints as the original FAUST circuit.

5.2.2 DSPIN Clock Tree

On the Front-End phase, we have added a buffer or an inverter on the clock input of each DSPIN router. These buffers/inverters are used to support the clock tree synthesis following the GALS approach. The construction of the DSPIN clock tree follows four steps (Figure 5.6):

1. The buffer/inverter on the clock input pin of each DSPIN router is manually placed in the middle of the area occupied by the cluster. This placement is done with the floorplanning tool. Thus, the wires of the each DSPIN router clock tree are as short as possible.
2. A clock tree is synthesized for each DSPIN router. The starting point of the clock-tree is the buffer/inverter on the clock input pin of the router. Therefore, the *Clock_root_pin* variable is set to identify the starting point, the root pin, of the clock-tree. Each clock tree is synthesized with 5% skew target.
3. Once these clock trees synthesized, each clock tree is characterized with its input delay, its skew, and its input capacitance. Therefore, the *Macromodel* of the clock tree is extracted for the next step.
4. Finally, a top clock tree is synthesized to balance the clock trees of all the DSPIN routers. Following the GALS approach, the top clock tree is balanced with a 30% skew while the leaves have a 5% skew. To avoid modifying the skew of the bottom clock trees, it is mandatory to preserve integrally the bottom clock trees. Therefore, the root pins of the bottom trees are tagged with the *PreservePin* tag. Hence, the clock-tree synthesis tool can only balance the top clock tree and does not modify the cells beyond the pin tagged with the *PreservePin* statement. The *Macromodel* obtained in previous step is used to characterize the bottom clock trees because they are hidden by the *PreservePin* statement. At the end of the synthesis, the top clock tree is balanced with 30% skew while the bottom

has 5% skew. As a result, the communications between routers are mesochronous.

After synthesis, each bottom clock tree contains from 210 to 307 buffers/inverters, depending on the area covered by the DSPIN router. The higher the area covered, the higher the balancing effort, and the higher the number of buffers/inverters required. As example, the router on position (1,2) requires 307 buffers/inverters as its area is the highest of all of them.

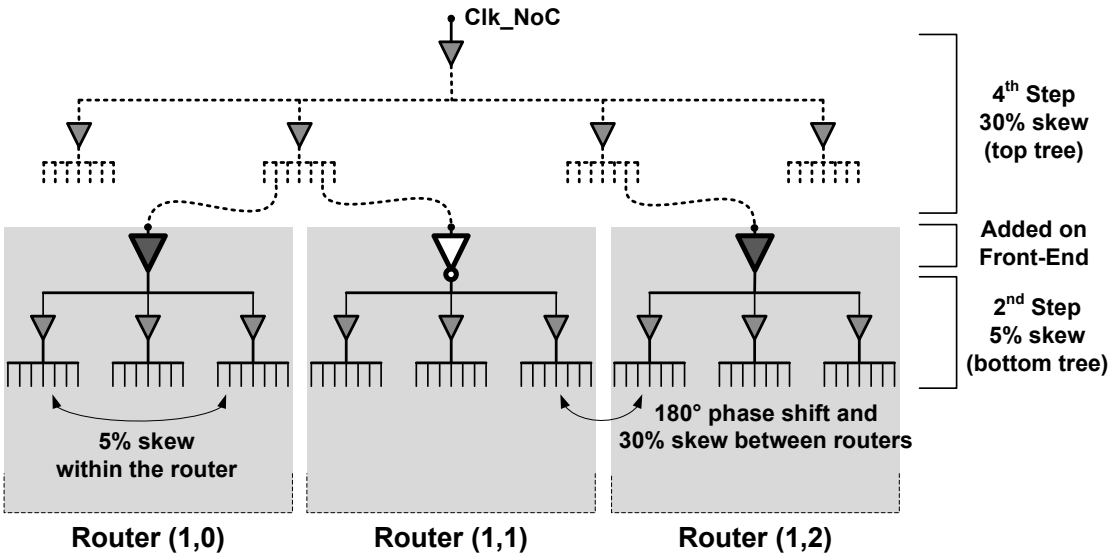


Figure 5.6 DSPIN clock tree

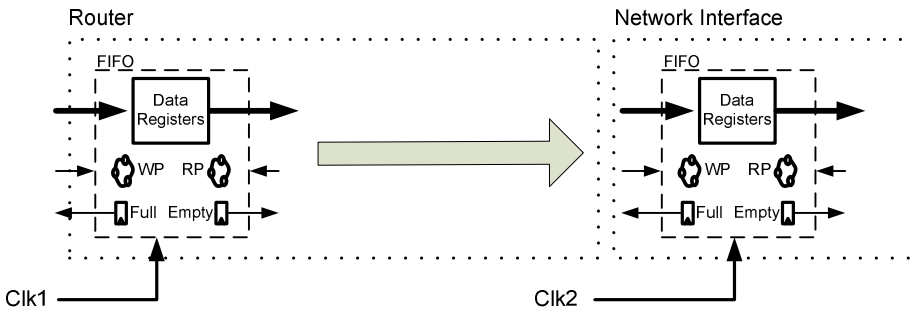
The top clock tree contains 69 buffers/inverters and its skew is around 1000ps. Its skew is 30% for a clock period of 3.4ns. The targeting skew before synthesizing was 40% and a maximum transition time of 450ps. The obtained clock tree has 30% skew (instead of 40%) and a maximum transition time of 430ps.

The clock-gating cells have been accepted by the clock tree tool. Moreover, to maximize the power saving of the clock-gating technique, these cells have to be as close as possible of the clock root pin. Hence, the power saving is achieved on the sequential cells and on the clock tree buffers. Therefore, the clock-tree synthesizing tool was properly configured to move these cells. In SoC Encounter, this option is called *PadBufAfterGate*, which means, that the padding buffer cells are placed after the clock-gating cell instead of before.

5.2.3 Mesochronous and Asynchronous Links

The communication between neighbors routers are mesochronous as the clock tree is not equilibrated between routers. Moreover, the communications between routers and subsystems are fully asynchronous because they use different clock frequencies. The bi-synchronous FIFO, interfaces the mesochronous/asynchronous interfaces without complex back-end flow. Just a timing constraints file has to be properly set to guarantee a correct tool implementation.

- For the **asynchronous interfaces**, the *set_false_path* condition is set between the clock signals of different clock frequency. Hence, the tool understands the asynchronous nature of this kind of interfaces. Otherwise, the tool tries unsuccessfully to synchronize non-synchronous interfaces while the synchronization is done by the bi-synchronous FIFO. Figure 5.7 shows the declaration of the *set_false_path* condition between a router and the network interface.
- For the **mesochronous interfaces**, a *set_multi_cycle_path* condition is added on the output ports of the FIFO data registers. This condition informs the tool that the content of the FIFO data registers are not written and read on the same clock cycle. The writing and later reading of bi-synchronous FIFO data register is delayed by the synchronization latency. Hence, the data is stable when it is read, the timing paths are simplified and the tool can easily interface the mesochronous interface. Figure 5.8 shows the declaration of the *set_multicycle_path* between the mesochronous communication of two routers.



set_false_path -from [Clk1] -to [Clk2]
set_false_path -from [Clk2] -to [Clk1]

Figure 5.7 Timing constraints for asynchronous interface

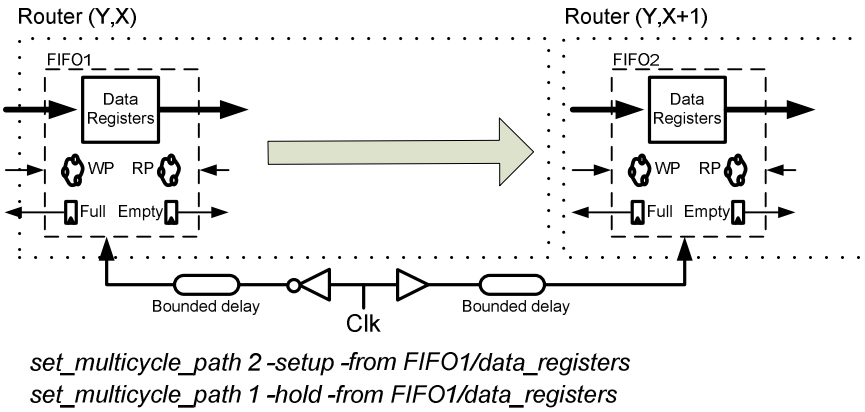


Figure 5.8 Timing constraints for mesochronous interface

5.3 Implementation Validation and Parameter Extraction

In this section, the physical implementation is firstly analyzed with detailed static timing analysis tool in order to estimate the maximum clock frequency. Later, the gate-level netlist is simulated with the back-annotation RC delays. A real-data simulation application is used to validate the correctness of the netlist. Finally, the power consumption of the DSPIN NoC is estimated using a back-annotated simulation of the real application.

5.3.1 Maximum Operating Frequency

Once the Back-End flow is fully developed, the gate-level netlist and the timing file *sdf* is extracted. The static timing analysis with detailed RC parasites is used to determine the maximum operation frequency of the circuit.

Both implementations are analyzed, the one without clock-gating and the one with clock-gating. Table 5.1 presents the maximum operating frequency for the DSPIN NoC and for the FAUST sub-system IPs. Worst-case conditions analysis is used. The clock-gating implementation obtain better performances due to simplified critical path and lower area, thus reducing the wire congestion.

Table 5.1 Maximum operating frequency on worst-case conditions

	Maximum DSPIN frequency	Maximum FAUST IP frequency
Without clock-gating	274 MHz	157 MHz
With clock-gating	289 MHz	157 MHz

5.3.2 Back Annotation Simulation

The gate-level netlist and the timing file *sdf* are used to simulate with ModelSim the whole circuit with the back-annotation information. Thereby, the circuit is functionally verified with the real application data. This kind of simulations is very accurate as the wire/gate delays are taken into account on the simulation. Therefore, the simulation executes slowly and requires powerful machines.

Both implementations, with and without clock-gating, are successfully simulated and validated for a full ODFM frame demodulation. The circuit is tested with different clock frequencies to verify the correct operation of the mesochronous and asynchronous interfaces, and GALS approach. Towards that end, the simulator is properly configured to avoid detecting non-desirable setup time and hold time violations. These false violations came from three sources:

- **Synchronizers**: The main task of the synchronizers on the bi-synchronous FIFO is to interface independent frequency/phase clock domains even on a setup or hold time violation.
- **Reset synchronization**: On each DSPIN router, a Reset synchronizer was added to guarantee a clear reset signal. The reset signal is proper resynchronized.
- **Invalid data:** Writing data on the FIFO when it is empty can induce a setup or hold violation on the read side. However, this violation should not be ignored, as the read data is invalid.

For these three sources of violations, a *tcheck_set* statement is used to avoid detecting them and stopping the simulation. This statement disconnects timing checks on the indicated device.

5.3.3 Power Consumption Analysis

The power consumption is analyzed with the PrimePower tool. The real activity of the circuit is used on the power analysis to maximize the accuracy of the results. Therefore, a back-annotation simulation of the circuit is performed on the gate-level netlist of the circuit to extract the switching activity of the DSPIN routers. The simulation performs a full OFDM frame demodulation. The simulation of 1500μs takes 3 days on a 64-bit Opteron 2.2GHz 8GB RAM, and generates a stimuli file of 32-GBytes.

In order to estimate the real power consumption of the circuit, the back-annotated RC parties are extracted for typical operating conditions. The DSPIN routers and clock-trees power consumption are extracted for both implementations the one

without clock-gating and the one with clock-gating. The detailed power consumption of both implementations can be found in Appendix C.

5.3.3.1 Without clock-gating

For the physical implementation without clock gating, the power consumption of the router do not depends so much on the router activity. The power consumption per router is 9mW at 149MHz and 14 mW at 274MHz. The registers consume about 86% of the total power. On the other hand, the clock tree distribution per router consumes 7.6mW at 149MHz.

The total power consumption of the DSPIN routers and the mesochronous clock tree is 317 mW at 149 MHz (the NIC power consumption is not included). These results motivate us to implement a clock gating technique.

5.3.3.2 With clock-gating

With the clock-gating technique, the power consumption of the router is reduced. Table 5.2 summarizes the power consumption of an inactive router, a medium active router, and the highest active router in the implemented platform for two clock frequencies 149 MHz and 289 MHz. The clock-gating technique reduces the power consumption of the DSPIN routers by 67% at 149 MHz.

Table 5.2 Power consumption of DSPIN router

	149 MHz	289 MHz
Inactive router	2.06 mW	3.72 mW
Medium active router	3.00 mW	4.93 mW
Highest active router	4.23 mW	6.40 mW

The power consumption of the two FIFOs contained in the NIC is extracted for 149MHz and 289MHz. Table 5.3 resumes the power consumption of the FIFOs in the NIC in function of the activity of the NIC.

Table 5.3 Power consumption of FIFOs in the NIC

	149 MHz	289 MHz
Inactive NIC	0.24 mW	0.46 mW
Medium active NIC	0.52 mW	0.74 mW
Highest active NIC	0.91 mW	1.33 mW

Table 5.4 shows the clock-tree power consumption for two clock frequencies. The *Top clock tree* corresponds to the mesochronous clock tree, while the *Bottom clock tree* corresponds to the synchronous DSPIN route clock tree.

Table 5.4 DSPIN clock-tree power consumption with clock-gating

	Power consumption at 149 MHz	Power consumption at 289 MHz
Top clock tree	1.23 mW	2.39 mW
Bottom clock tree	47.70 mW	92.30 mW
Total	48.93 mW	94.69 mW

Table 5.5 shows the total power consumption with clock-gating for 149 MHz and 289 MHz when the read OFDM demodulation application is simulated. Around 50% of the total power consumed is consumed by the clock-tree. Comparing with the non clock-gating implementation, this implementation saves 67% of power consumption at 149 MHz.

Table 5.5 Total power consumption with clock-gating

	Power consumption at 149 MHz	Power consumption at 289 MHz
DSPIN routers	54.25 mW	92.17 mW
FIFOs on NIC	9.96 mW	14.92 mW
Clock tree	48.93 mW	94.69 mW
Total (all routers)	113.14 mW	201.78 mW
Total (per router)	5.65 mW	10.08 mW

5.3.3.3 Summary

The utilization of the clock gating technique allows to reduce drastically (67%) the power consumption of the DSPIN architecture while improving its maximum clock frequency (from 274 MHz to 289 MHz). The power optimization of the FIFOs in the NIC is also achieved. Its power consumption is reduced from 0.91 mW to 0.24 mW when in idle state.

On the other hand, the power consumption of the clock tree is still high, even when the clock gating technique reduces it by 67%. The clock-tree power consumption is as high as the DSPIN router one. This is the consequence of a circuit highly dominated by registers. The DSPIN router just forwards the data between input and output ports; it does not perform combinational operations with the data.

5.4 DSPIN versus ANOC Comparison

The initial FAUST implementation was originally build around a fully asynchronous network-on-chip called ANOC, and designed by CEA-Léti. This NoC is replaced by the multi-synchronous DSPIN NoC while preserving the same chip area and pad-ring. In this section, the physical implementation of DSPIN and ANOC are compared in terms of area, throughput, packet latency, power consumption, and programmability.

5.4.1 Area

The ANOC router was implemented as a hard macro. Its area is 0.21mm² with a cell density of 95%. The *GALS_interface* module was implemented as a soft macro and its area is computed assuming 95% of cell density. On the other hand, DSPIN is implemented as a soft macro and no area is reserved for the router (a 95% integration density is assumed). Just some regions are defined to condition the placement tool. Taking into consideration that the DSPIN router requires a clock tree while ANOC does not, the area comparison is done on the total area including the clock tree. The DSPIN router area after place and route is 0.153mm² while its clock-tree area is 0.0015mm². Assuming 95% of cell density, the DSPIN router area is 0.161mm² while the clock-tree area is 0.0016mm². Table 5.6 summarizes the area comparison between ANOC and DSPIN NoCs. The total DSPIN area is 33% smaller than the ANOC area.

Table 5.6 Area comparison between ANOC and DSPIN NoCs

	ANOC router	DSPIN router
Router	0.211 mm²	0.161 mm²
Interface GALS	0.070 mm²	0.024 mm²
Clock tree	0.000 mm²	0.0016 mm²
Total	0.281 mm²	0.187 mm²

5.4.2 Throughput

The throughput on the ANOC router depends on the fabrication process, on the voltage applied, and on the temperature condition. For worst-case analysis at 1.08V, and 105°C, the throughput of ANOC is 160Mflit/s. In nominal process conditions, its throughput is 220Mflit/s. However, it has not been verified on the real FAUST circuit because the Synchronous-to-Asynchronous and Asynchronous-to-Synchronous interfaces limit the measure to 160Mflit/s. In principle, the asynchronous circuits have

the advantage to auto-adapt their performances to the process, temperature, and voltage of the circuit. In contrast, synchronous circuits have their clock frequency limited to the worst-case process to guarantee their operation for any fabrication process variation, any temperature condition, and a voltage range.

The DSPIN router throughput depends exclusively on its operation frequency. Its throughput is one flit per clock cycle (1Mflits/s for a clock frequency of 1MHz). The maximum operating frequency for the DSPIN router on worst-case analysis is 289MHz and 408MHz on nominal-case. Consequently, the DSPIN throughput is 289Mflit/s in worst-case and 408Mflit/s in nominal-case.

In terms of critical path analysis, the ANOC critical path crosses four times the long wires in between ANOC routers while DSPIN crosses just one time. This comes from the fact that ANOC uses a 4-phase QDI asynchronous protocol. Thus, the long wire delay has four times higher influence on the ANOC router rather than on the DSPIN router. Consequently, on deep submicron technologies where the interconnect delays will be higher than the gate delays, a multi-synchronous architecture as DSPIN would have higher packet throughput than an asynchronous one as ANOC. Fortunately, pipeline stages can be inserted on the long wires in order to cope with these delays, despite of the added latency.

Table 5.7 shows the throughput comparison between the ANOC and DSPIN routers. On a real implementation, ANOC will operate on its nominal conditions 220Mflit/s while the DSPIN router should be clocked not far away from the worst-case condition 289MHz to improve the fabrication yield.

Table 5.7 Throughput comparison between ANOC and DSPIN routers

	ANOC	DSPIN
Throughput on worst-case conditions	~ 160Mflit/s	≤ 289Mflit/s
Throughput on nominal conditions	~ 220Mflit/s	≤ 408Mflit/s

5.4.3 Packet Latency

As seen on the DSPIN router section (Chapter 3), the routing latency is decomposed in three stages: first, intermediate and last router. This decomposition can be used to analyze the latency of any communication path.

The latency of the ANOC router depends on the fabrication process, on the voltage applied, and on the temperature condition. Real values have been measured

on the real implementation of the FAUST circuit. The computation of the intermediate latency is not conditioned by the operating clock frequencies of the subsystem. Its latency is 6.8ns on the real circuit. However, the latency of the first and the last router are conditioned by synchronous-to-asynchronous and asynchronous-to synchronous interfaces on the subsystems. Unfortunately, the first and last router latencies could not be measured separately.

DSPIN router latency depends exclusively on the router and subsystem operation frequencies. Its detailed calculation can be found in Chapter 3. Table 5.8 details the latency comparison for two clock frequencies 150MHz and 250MHz. For the analysis, the clock frequency on the ANOC router means the clock frequency of the subsystem. While the clock frequency on the DSPIN router means the clock frequency for the DSPIN router and subsystem. The intermediate router latency on the ANOC router is lower than the DSPIN one. This comes from the fact that the DSPIN router resynchronizes the data packets on each hop. To obtain the same intermediate router latency, the DSPIN router should be clocked at least 367MHz. Moreover, the first and last router latency is better optimized on the DSPIN side.

Table 5.8 Latency comparison between ANOC and DSPIN routers

	F = 150 MHz		F = 250 MHz	
	ANOC	**DSPIN**	**ANOC**	**DSPIN**
Intermediate router latency	6.80 ns	16.66 ns	6.80 ns	10.00 ns
First and last router latency	60.00 ns	56.66 ns	47.00 ns	34.00 ns

Table 5.9 shows the latency of the ANOC and DSPIN router for 5 and 9 hops path. It is clear that the ANOC router have lower latency than the DSPIN router for low clock frequencies, but the latencies are quit similar when the clock frequency increases.

Table 5.9 Latency analysis for 5 and 9 hops path

	F = 150 MHz		F = 250 MHz	
	ANOC	**DSPIN**	**ANOC**	**DSPIN**
Latency for 5 hops path	80.00 ns	106.66 ns	68.00 ns	64.00 ns
Latency for 9 hops path	106.66 ns	173.30 ns	96.00 ns	104.00 ns

5.4.4 Power Consumption

The ANOC router was implemented using STMicroelectronics standard cells and the TAL library [Maurin03]. Low power cells with High Vt (low leakage and low speed) and Low Vt (high leakage and high speed) were used to satisfy the required performances. Thus, the leakage power on the ANOC router cannot be neglected. Appendix C details the power consumption estimation per router on the ANOC estimation.

The power consumption comparison between the DSPIN and ANOC network-on chip is performed for 15 of the 20 FAUST routers because the other 5 are not used on the OFDM demodulation application and were not estimated on the ANOC implementation.

For ANOC, the power consumption of the 15 active routers is 31.04 mW where its leakage is 5.1 mW (0.37mW per router). The total power consumption of the 15 *GALS_interface* modules (see Appendix B) is 24.40 mW where its leakage is 3.6 mW.

For DSPIN, the power consumption of the 15 active routers, FIFOs and clock-trees are computed using the Appendix C results. Table 5.10 shows the detailed power consumption (router, FIFO, and clock-tree) for the ANOC and DSPIN.

Table 5.10 ANOC and DSPIN power consumption

	ANOC	DSPIN (149MHz)	DSPIN (289MHz)
Routers power	31.04 mW	43.38 mW	72.75 mW
FIFOs on GALS interf.	24.40 mW	8.40 mW	12.14 mW
Clock-tree power	0.00 mW	36.69 mW	71.01 mW
Total (for 15 routers)	55.44 mW	88.47 mW	155.90 mW
Total (per router)	3.69 mW	5.89 mW	10.39 mW

The power consumption of the ANOC router is lower than the one of DSPIN. This comes from the fact that the DSPIN uses larger FIFOs (7 words depth compared to 2 words depth on ANOC). On the other hand, the *GALS_interface* module on ANOC consumes higher than the DSPIN one, because the ANOC module contains 4 FIFOs while the DSPIN module contains just 2. Moreover, ANOC uses 4-phase handshake protocol on the FIFOs that is higher power consuming than a synchronous approach.

In terms of total power consuming, ANOC is less power consuming than DSPIN, even at 149 MHz. The power consuming of DSPIN increases almost linearly with the clock frequency. Moreover, 50% of the total power consumption of DSPIN is due to the clock-tree network. Thus, in a future version of DSPIN, a power management unit should clock-gate the full DSPIN router clock-tree when it is not active.

5.4.5 Programmability

From a programmability point of view, the address-based routing algorithm is more versatile than the source routing algorithm. Address-based routing algorithm can be easily implemented on stream-oriented and shared-memory architectures as demonstrated on the implementation of DSPIN in a stream-oriented platform.

On the other hand, source routing algorithm is not suited to shared-memory architectures because the network interface controller has to know all the possible packet destinations in order to route the packets. Consequently, the NIC becomes complex because it requires to store all the routing paths managed by the IP; thus, being difficult to reprogram dynamically. Moreover, when the routing path does not fit into the packet header (first flit), a path extension mechanism has be used; thus, increasing the NIC complexity.

Source routing algorithm is suited for stream-oriented architecture where communication graph is known and can be analyzed before mapping the application. Thus, it is possible to avoid congested links by choosing the routing path of some communication. In case of dynamic reallocation of new communications, it is required to know all the current communication paths in order to avoid the congested links. Therefore, a global path allocator should decide the best routing path for the new communications.

Generic architectures, where the task graph can be modified dynamically or the number of communications per IP is not limited to a reduced number, should be implemented on address-based routing algorithm. Firstly, the address-base can use the destination address of the packet to translate the address into the routing address without programming the NIC. Secondly, no path extension mechanism is required as the address-based routing is more compact than the source-routing algorithm.

5.5 Conclusion

A physical implementation of the DSPIN network-on-chip on the generic, stream-oriented, FAUST platform has been presented. The multi-million gates FAUST chip using DSPIN has been physically implemented up to mask layout to demonstrate the easily implementation of the multi-synchronous DSPIN NoC in an industrial flow. The Front-End and Back-End flows do not require custom tools. Commercial synthesis tools as Synopsys Design Compiler, and place and route tools as Cadence Encounter are suited to implement this architecture.

We demonstrated that the multi-synchronous DSPIN architecture can be simply and automatically implemented. The floorplanning of a SoC with DSPIN NoC is very flexible due to the DSPIN soft macro conception. It just requires defining 5 regions per router on the sides of the cluster. The mesochronous and asynchronous interfaces are easy implemented thank to the, correct by construction, bi-synchronous FIFO. A simple timing constraints file guarantees the correct implementation of these interfaces. The mesochronous clock-tree distribution network has been automatically implemented following four steps. Therefore, the intra-cluster clock skew is lower than 5% while the inter-cluster clock skew can reach 30%. The exclusion of asynchronous and custom cells in the DSPIN architecture simplifies the implementation flow and allows implementing it on fully synchronous Front-End and Back-End flows. Moreover, the architecture can directly ported to other CMOS process technologies, as it is fully synthesizable.

We have compared the ANOC and DSPIN implementations on the same FAUST platform. Both implementations use the same process technology and has the same die area. DSPIN is 33% smaller than ANOC, and has 31% higher throughput than ANOC. In terms of packet latency, DSPIN has predictable packet latency as it depends on the clock frequency, while ANOC latency depends on the process, temperature, and voltage. Both architectures have similar packet latencies when the clock frequency is higher than 250 MHz; otherwise, ANOC has lower latency. The maximum operating frequency for DSPIN is 289 MHz on worst-case analysis. In terms of power consumption, ANOC consumes less power than DSPIN, even at low frequencies (150 MHz). The clock-gating implementation on DSPIN reduced its power consumption by 67%; however, it is still higher than the one of ANOC. The DSPIN clock-tree consumes as much power as the DSPIN router itself.

We have analyzed the advantages of the address-based algorithm over the source-routing algorithm in a generic architecture. Address-based algorithm is more

generic and can be easily implemented on shared-memory and stream-oriented architectures. The algorithm is more compact in number of bit and does not require path extension mechanism to reach long distance paths. Source routing algorithm is suited to stream-oriented architectures where the task graph of the application is known before mapping the application; thus, the routing path can be optimized for the application and the congested links can be avoided. For generic and dynamic reallocation of the task graph, address-based algorithm is more suited than source routing algorithm.

Chapter 6

Conclusion

The experience gained in the design and physical implementation of the 32 ports SPIN network was precious to define a new architecture well suited to the Globally Asynchronous, Locally Synchronous (GALS) paradigm. This architecture is the DSPIN Network-on-Chip proposed by Alain Greiner at the University of Pierre et Marie Curie. However, this architecture did not provide guaranteed service traffic, and it has not been physically implemented.

In this thesis, we addressed the following issues: We introduced and evaluated a low-cost guaranteed service mechanism using virtual channel in the DSPIN architecture. We designed an efficient and robust bi-synchronous FIFO able to interface synchronous systems. Finally, the DSPIN architecture has been physically implemented on a multi-million gate System-on-Chip.

6.1 Guaranteed Service

We have demonstrated the implementation of the guaranteed service in the DSPIN architecture. The virtual channel approach (generally used to multiplex several logical channels on the physical link between routers), can be applied to the router itself, making TDM multiplexing possible in a GALS, clustered multiprocessor architecture. With this low cost method, the DSPIN architecture provides the system designer hard bounds for both the latency (upper bound) and the throughput (lower bound) of a limited number of point-to-point communications. The overhead of the virtual channel increases the area by 43%, while the number of channels has been doubled. The router and network-interface controller architectures have been analyzed in detail. Modifications of the state machines have been proposed in order

to increase/decrease the guaranteed service throughput without provoking starvation situations on the best effort traffic.

The strictly bounded latency and throughput result from the following choices:

- DSPIN router uses a deterministic deadlock free routing algorithm.
- DSPIN uses separate sub-networks for request and response traffic.
- The storage elements for BE and GS traffic are independents.
- The shared resources between BE and GS traffic are only combinational logic and long-wires.
- The virtual channel allocation policy is fair and does not have starvation situation.
- The NIC has independent ports for BE and GS traffic.

Therefore, the implemented levels of the OSI reference model, from transport to physical level, are deadlock free without starvation situations. The latency of the guaranteed service traffic is predictable and bounded. Likewise, the throughput of the guaranteed service is guaranteed up to 50% of the channel bandwidth and can be increased/decreased by modifying the state-machine allocation policy. In order to achieve these guarantees, the DSPIN routers are clocked with the same clock frequency, but a clock skew can exist between neighbor routers. Thus, DSPIN uses a low power mesochronous clock-tree distribution compatible with the GALS approach. The communication between independent clock domains is carried out by bi-synchronous FIFOs. Finally, neighbor routers have inverted clock phases in order to avoid metastability failures in the bi-synchronous FIFOs.

The original DSPIN architecture has been updated in many aspects other than the guaranteed service traffic. The packet format has been redefined in order to respect the OSI reference model. The network clock frequency is independent of the system clock frequency. The architecture is synthesizable with standard cells only, without asynchronous or custom cells. Power reduction techniques have been designed and introduced in the architecture. The mesh topology, associated with the distributed implementation of the router itself solves the problem of long wires.

The DSPIN architecture has been simulated on a 10x10 cluster platform to evaluate its saturation threshold. On the other hand, we have analyzed the tradeoff between FIFO depth, router area, and architecture performance. The optimum performance at minimum FIFO depth and router area is a packet length shorter than 10 flits and the BE FIFO depth of 7 words. Finally, the estimated silicon area for the DSPIN router is 0.057mm² on CMOS 90nm process. The clock frequency is 500MHz. The synthesis is successful up to 833MHz while its area increases by 37%.

6.2 Synchronization

We have designed a bi-synchronous FIFO able to interface two synchronous systems with independent clocks. The design is synthesizable with standard cells only, and does not use asynchronous or custom cells. The FIFO uses a new encoding algorithm, called bubble encoding, which has been demonstrated to have a better density than previous designs with similar performance. A smart definition of the FIFO pointers avoids using status registers. Thus, the write and read pointers are directly combined to obtain the Full and Empty signals. Moreover, the architecture uses simple registers with classical scan chain path, in order to avoid using a hard macro RAM memory and its associated BIST testing technique. The bi-synchronous FIFO can be used for a fully an asynchronous interface (different clock frequency and phase) or a mesochronous interface (same frequency but different phase). Through optimization, the FIFO latency for a mesochronous interface may be reduced. Its latency is 2-3 clock cycles in asynchronous mode, and 1-2 clock cycles in mesochronous mode. In terms of throughput, the FIFO can deliver 100% throughput when its depth is six or more words.

We have synthesized a 32-bit 8-word deep bi-synchronous FIFO on CMOS 90nm technology, and estimated its maximum operation frequency at 1GHz and its area at 6581μm^2. Finally, the proposed design has performances comparable to the equivalent best-known design, and it has a silicon area 33% smaller than the Gray-code FIFO generally used in industry.

The bubble encoding and the architecture of bi-synchronous FIFO have been patented by STMicroelectronics.

6.3 Physical Implementation

From the physical implementation point of view, we have demonstrated that the DSPIN architecture can be integrated in an industrial design flow based on state-of-the-art commercial tools.

The stream-oriented FAUST platform has been chosen as the SoC implementation. The full physical implementation flow has been performed using commercial tools and automated scripts. Thank to the modularity of the DSPIN router and its flexible implementation (synthesizable soft macro), the floorplanning of complex SoCs is simplified.

The NoC of FAUST (ANOC) has been replaced by the DSPIN NoC without modifying either the FAUST architecture or the chip area. A hierarchic synthesis has been performed on the Front-End flow. The floorplanning of the FAUST chip with the DSPIN NoC has been performed with minimum floorplan modifications in order to prove the adaptability of the DSPIN architecture in a SoC implementation. The mesochronous clock-tree has an intra-cluster clock skew lower than 5%, while the inter-cluster clock skew can reach 30%. The clock gating technique has been introduced to reduce by 67% the power consumption of the DSPIN NoC without reducing its performances. The asynchronous and mesochronous interfaces have been easily implemented thanks to the, correct by construction, bi-synchronous FIFOs and by simple timing constraint directives. The maximum DSPIN clock frequency on the FAUST platform for 130nm process is 289 MHz (worst-case).

We have compared the DSPIN and ANOC implementations on the FAUST platform. In terms of area, DSPIN is 33% smaller than ANOC. DSPIN has 31% higher throughput than ANOC. In terms of packet latency, DSPIN has predictable packet latency as it depends on the clock frequency, while ANOC latency depends on the process, temperature, and voltage. Both architectures have similar packet latencies when the clock frequency is higher than 250 MHz; otherwise, ANOC has lower latency. In terms of power consumption, ANOC consumes lower power than DSPIN even at low frequencies (150 MHz). The DSPIN clock-tree consumes as much power as the DSPIN router itself. Consequently, DSPIN is optimized for low area and high performance architectures, while ANOC is optimized for low latency and low power application.

We have also analyzed the programmability of an NoC using address-based routing and one source-routing. Address-based routing is more generic and versatile than source routing because it can be used on shared-memory architectures as well as on source-routing architectures. The NIC for an address-based routing is simpler than for source routing. This comes from the fact than on a shared-memory, the destination address can be easily recoded into an address-based address by simply taking the MSB bits. On the other hand, a complex Look-up Table has to be implemented on the NIC to convert addresses into routing paths for source routing. On a message-passing architecture, the address-based routing information is more compact than the source routing one, thus increasing the packet payload. Moreover, address-based routing does not require complex path-extension mechanisms compared to source-routing when the routing path does not fit into a single flit.

The top-down design and physical implementation of the DSPIN architecture confirmed the objectives defined at the beginning of this thesis, which were to design a distributed and synthesizable NoC with a flexible and simple industrial implementation flow suited to the GALS approach.

6.4 Answers to the Open Questions

In the first chapter, we formulated some open questions that the state-of-the-art could not answer completely. In this section, we summarize the answers given by our work:

6.4.1 Quality of Service

- **Packet latency**: DSPIN guarantees hard bounds on the packet latency of the GS traffic for a limited number of communications (when no path conflict exists). This latency is deterministic and depends on the network clock frequency.
- **Throughput**: DSPIN guarantees at least 50% throughput to the GS traffic for a limited number of communications (when no path conflict exists). This guarantee is hard bounded and can be easily increased by modifying the allocation state-machine.
- **Overhead**: The area increase of the DSPIN router with GS+BE traffic compared to a DSPIN router with only BE traffic is of 43%. Doubling the number of channels costs less than a 50% area increase.
- **Shared resources**: DSPIN architecture uses virtual channels with a buffer per channel. The shared resources on DSPIN architecture are the long wires (intra-cluster wires). No data register is shared between BE and GS traffic.
- **Path allocation**: The guaranteed service traffic must be allocated by a central path allocator. No hardware implementation is designed as the allocation and reallocation is not frequent. Consequently, a software task manages a data-graph of all current allocated paths on the SoC, and it decides the allocation of new traffics.
- **GALS**: DSPIN is suited to the GALS approach. Each cluster can be considered as an independent clocked island with its own clock frequency. Moreover, the NoC uses a mesochronous clock-distribution network, which can be unbalanced up to 50% skew.

6.4.2 Synchronization

- **Latency**: The latency of the bi-synchronous FIFO is 2-3 clock cycles in asynchronous mode, and 1-2 clock cycles in mesochronous mode.
- **Throughput**: The bi-synchronous FIFO is able to deliver 100% throughput. Its depth must be 6 words or more in asynchronous mode and 5 words or more in mesochronous mode.
- **Robustness**: The robustness of the bi-synchronous FIFO is achieved by two-flop synchronizers. The synchronizers can be chosen from the standard cell vendor library to improve the robustness.
- **Process, temperature, and voltage variation**: The bi-synchronous FIFO is robust to process, temperature, and voltage variations because the communications between independent clock domains is synchronized by two-flop synchronizers. On mesochronous mode, the interface is less robust because some two-flop synchronizers are replaced by one-flop synchronizer in order to reduce the latency.
- **Portability and industrialization**: The bi-synchronous FIFO is suited to industrial implementation. It contains neither asynchronous nor custom cells. Moreover, its physical implementation is flexible because no hard macros such as RAM memories are used.
- **Testability**: The bi-synchronous FIFO can be tested by classic test methodologies such as scan paths.
- **Density**: Its area is 6581 μm^2 for 8-word 32-bit bi-synchronous FIFO working at 1GHz clock frequency on CMOS 90nm technology.
- **Flexibility**: The bi-synchronous FIFO is floorplan flexible. It does not contain hard macros. Its unique condition is to place the two-flop synchronizers as close as possible to increase robustness with respect to metastability.

6.4.3 Physical Implementation

- **Soft macro**: DSPIN is physically implemented as a soft macro. It does not contain any hard macro.
- **Floorplanning**: The DSPIN router is composed of 5 modules, witch have to be placed on the sides of the cluster in order to be near the neighbor router modules. In terms of timing constraints, simple timing constrains are used in order to indentify the mesochronous and asynchronous interfaces.

- **Industrialization**: DSPIN is suited to be implemented on an industrial flow. Commercial tools from the Front-End to the Back-End synthesis can be used. Neither custom tools nor spice simulations are required to design and verify the correct operation.
- **Portability**: DSPIN is fully portable to any standard cell technology. It does not contain asynchronous or custom cells, or RAM memories. Its architecture is fully synthesizable with industrial standard cells.
- **Clocking**: DSPIN clock tree uses a mesochronous approach. Its implementation is automatic and it follows a bottom-up approach.
- **GALS**: DSPIN is suited to the GALS approach. Each cluster can be physically implemented as a stand-alone module and later assembled on the SoC. The DSPIN NoC manages the inter-cluster communications.
- **Clock boundaries**: The clock boundaries are implemented using bi-synchronous FIFOs, which are correct by construction. Thus, its physical implementation is simple to realize using commercial tools.
- **Power**: The clock gating technique has been implemented in DSPIN NoC. It reduced the power consumption by 63% compared with a non-clock gating implementation.
- **Long wires**: The only long wires in DSPIN architecture are the intra-cluster wires. Thus, the timing closure is simple because the long wires are restrained inside the cluster, which is an isochronous island.
- **Predictability**: DSPIN architecture is predictable because its throughput and latency depend on the clock frequency and all the routers use the same clock frequency.

6.5 Weakness

The distributed router architecture of DSPIN has many advantages, but there is one known drawback associated to the distributed approach, related to the power gating implementation.

Low power System-on-Chip uses different power reduction techniques as the clock-gating and power gating. The power-gating technique is a power reduction method that cuts down the power source of unused modules. Therefore, the gated modules consume neither static nor dynamic power. The entire gated module is

confined into a limited area, which has an independent power domain. Figure 6.1 shows an example of three independent power domains.

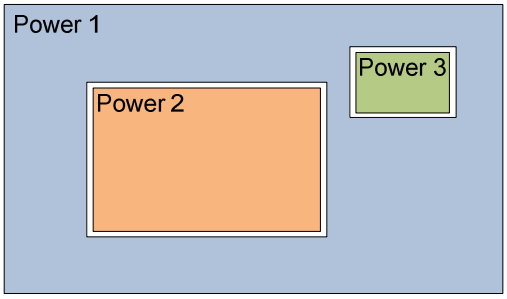
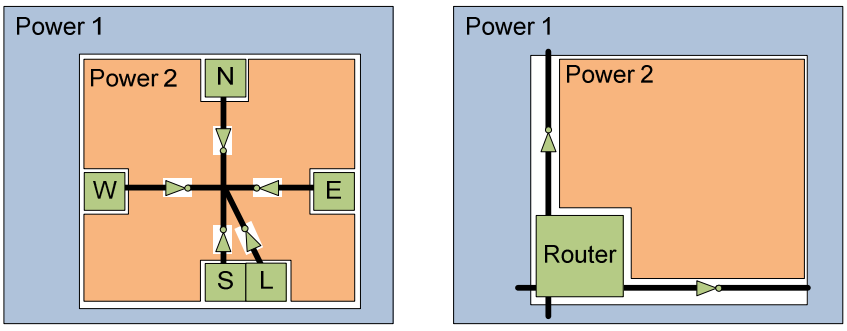


Figure 6.1 Power domains

When a distributed NoC is implemented on a power-gated architecture (one router per module), the power domain of the router has to be independent of the power domain of the gated module domain in order to allow the communications to pass through the router even when the module is gated. However, the distributed implementation of the DSPIN router itself makes this power gating difficult: The DSPIN router is not a well-identified hard macro-cell. It is split in five components (North, South, East, West and Local modules) that are placed by an automatic place & route tool. Moreover, the repeaters (buffers) of the DSPIN intra-cluster wires, being routed over the subsystem's module, have to be permanently powered in order to continue to operate. Figure 6.2a shows an example of DSPIN implementation where the N, S, E, W, and L modules, and repeaters should have an independent power domain (depicted in green). Moreover, the repeater interconnecting these modules has the same power domain as the DSPIN router.



a) DSPIN power domains b) Generic NoC power domains

Figure 6.2 DSPIN and generic NoC power domains

Figure 6.2b shows an example of a generic NoC implemented as a hard macro. The router has an independent power domain of the subsystem. The inter-router wires have a reserved path and a reserved power domain in order to power the long-wire repeaters. Therefore, the Back-End implementation is easier as the power domains are topologically disjoints.

A possible solution to this issue is to follow the same implementation approach as used on the retention-registers. Such registers have a flip-flop and a retaining latch built in a single library cell. The flip-flop can be power-gated while the latch is always powered in order to maintain the register data when the system is power gated. Therefore, the repeaters could be continuously powered by the same power rails as the retention registers using Always-On Buffers (AOB).

6.6 Future Work

After analyzing the results obtained in this thesis, the future work should be oriented in three main directions:

- **Power consumption optimization**: The clock tree power consumes 50% of the total power consumption. A clock gating technique has to be implemented at system level in order to maximize the reduction of the clock-tree power consumption when the router is in idle state.
- **Multiplexing Guaranteed Service traffics**: The multiplexing of GS traffics on the same link, while guaranteeing each communication, should be analyzed following the work of Kees Goossens in Æthereal [Rijpke03] [Gangw05] [Hans07].
- **Fault tolerant systems**: The network has to be able to route the packet on a NoC where some links are unavailable due to faults or to inexistent connections. Moreover, the algorithm must be low cost and simple. The research on X-Y routing algorithm with minimal routing tables is a low cost and good candidate for future fault tolerant NoCs. The work of Evgeny Bolotin et al. in [Bolotin07] has to be analyzed.

Appendix A

Synchronization Techniques

This chapter describes the state of the art on bi-synchronous FIFOs, in the context of Network-on-Chip using GALS approach.

A.1 Gray-Code FIFO

The Gray-code is a cyclic binary encoding algorithm where two successive values differ in only one digit. Thus, a Gray-code counter can be synchronized to another clock domain using a parallel synchronizer (Figure 1.18) without introducing errors. The value obtained after synchronization, can be the actual value or the previous value of the Gray-code counter. Even if a metastable event is present on the synchronizer, the synchronized values are limited to these two possibilities. In conclusion, Gray-code algorithm is suited to synchronize the value of a Gray-code counter.

A.1.1 Architecture

The Gray-code can be used to build RAM-based b-synchronous FIFOs. The architecture is composed of a RAM memory, a *write* pointer, a *read* pointer, a Full detector, and an Empty detector as shown in Figure A.1. The RAM memory is used to store the data to exchange from the write side to the read side. The *write* and *read* pointers determines the write and read address in the RAM. The *write* and *read* pointers are encoded in natural binary code. The Full detector compares the value of the *write* pointer to the *read* pointer to determine the fullness of the FIFO. Likewise, the Empty detector compares the *write* and *read* pointers to determine the emptiness of the FIFO.

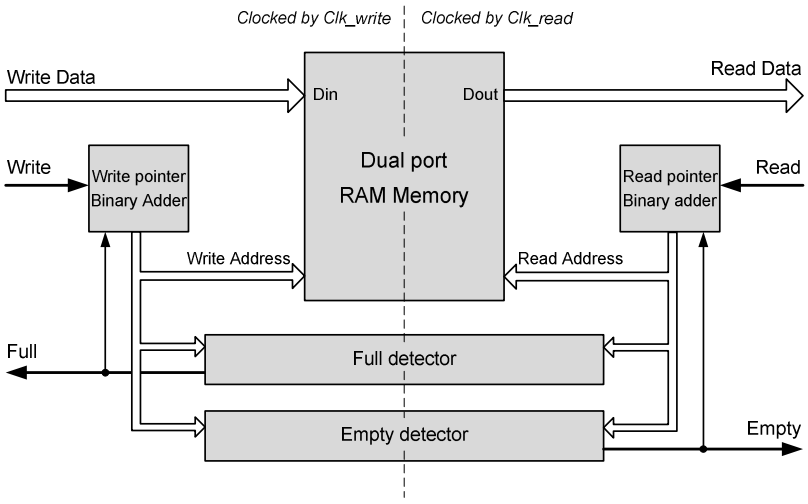


Figure A.1 Gray-code FIFO

In order to avoid catastrophic failures due to metastability, the Full and Empty detectors use the Gray-code to synchronize the pointers between the independent clock domains as shown in Figure A.2.

- **Full detector**: The *read* pointer is recoded to Gray-code, and then synchronized with a parallel synchronizer to the write clock frequency. Once synchronized, it is recoded to natural binary code to subtract its value from the *write* pointer. The result of the subtraction is used to assert the Full signal before overflow the FIFO.
- **Empty detector**: The *write* pointer is recoded to Gray-code, and then synchronized with a parallel synchronizer to the *read* clock frequency. Once synchronized, it is recoded to natural binary code to subtract its value from the read pointer. The result of the subtraction is used to assert the Empty signal before underflow the FIFO.

The subtracting devices on the Full and Empty detectors can be designed to detect full/empty conditions or quasi-full/quasi-empty conditions. Therefore, it is possible to anticipate the full/empty condition and for example stop the producer/consumer clock instead of just stopping the producer/consumer data.

The subtraction between the *write* and *read* pointers on the Full and Empty detectors could be done using Gray-code subtractors instead of natural binary code subtractors. Thus, the Gray-to-Natural converter could be omitted. However, the natural binary code subtractors are more area optimized than the Gray-code ones.

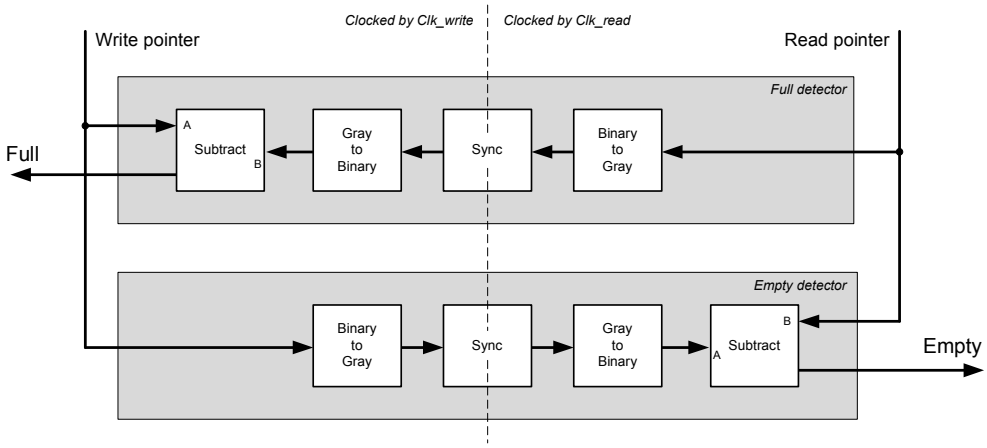


Figure A.2 Gary-code FIFO Full and Empty detectors

The RAM memory module can be replaced by registers. Thus, a Gray FIFO can be delivered as a soft macro without using a hard macro RAM. Moreover, the registers can be tested with scan chain strategy rather than a Built in Self-Test (BIST) for the RAM memory. On the other side, the integration density of a RAM memory is higher than the density of discrete registers. Thus, for a small FIFOs, a register-based memory is suited.

An example using RAM-based Gray-code FIFOs is the AsAP processor. It contains 32-word 16-bit FIFOs on TSMC CMOS 0.18 μm technology. Its area is 25000 μm^2. Its maximum operating frequency is 580MHz at 1.8V. Its power consumption is 10.3mW at 580MHz under maximum throughput operations. Detailed description of the FIFO can be found in [Apper07] and the AsAP processor in [Yu06][Yu07].

A.1.2 Analysis

The synchronization of a Gray-code counter is suited to interface a pointer between two independent clock domains. A FIFO implementing this technique is robust to the metastability failures. Moreover, the robustness of the FIFO can be increased by increasing the depth of the synchronizers. In this case, the latency of the FIFO is increased while being able to achieve 100% throughput.

The Gray-code FIFO architecture is suited to design deep FIFOs because the Gray-code is very compact compared to one-hot encoding. Example, the write and read pointers for a 256 words FIFO can require 8 bits and thus, the synchronizers require 8 bits. In the other hand, if one-hot encoding were used, it would be required 256-bit pointers and 256-bit synchronizer.

A RAM-based FIFO can be easily implemented on a Gray-code FIFO because the *write* pointers are encoded using a $\log_2$ algorithm as the Natural binary code. Thus,

the *write* and *read* pointers are directly the write and read address of the RAM memory. In the other hand, a RAM-based FIFO requires a testing methodology for the RAM memory. It can be a Built in Self-Test (BIST) or functional patterns to test the integrality of the FIFO. Thus, RAM-based FIFOs are suited for deep FIFOs. Moreover, a RAM memory is physically implemented as a hard macro. Thus, the floorplanning of the chip with RAM-based FIFOs became more constrained due to the hard macro blockages (width, height, and metal levels used), the need of manually place these RAM memories, and to modify the position of these memories when the wire congestion or timing constrains are not satisfied.

Finally, the conversion to Gray-code and later reconversion to Natural binary code, and the subtraction devices on the Full and Empty detectors requires complex logic schemes compared to other FIFOs as the one of J. Jex et al.

A.2 T. Chelcea et S. Nowick FIFO

T. Chelcea and S. Nowick (TCSN) propose a mixed-timing FIFO for synchronous-to-synchronous, asynchronous-to-synchronous, and synchronous-to-synchronous interfaces in [Chelcea04] and patented in [Chelcea07]. In this analysis, just the synchronous-to-synchronous FIFO is detailed due to the limited scope of the chapter.

A.2.1 Architecture

The TCSN FIFO is a register-based FIFO able to interface two synchronous systems where there is not relationship between their clock frequencies. The architecture view of the FIFO is depicted in Figure A.3. It is composed by a Full detector, an Empty detector, Data Cells (contain the write/read pointers and the stored data), a Put controller, and a Get controller. The number of Cells determines the depth of the FIFO.

The *write* and *read* pointers are encoded using the one-hot encoding algorithm. The pointers are distributed on the Cell elements, a 1-bit register per Cell element and per pointer. Figure A.4 details a Cell element. The top left register is the *write* pointer register while the bottom left register is the *read* register. In the middle of the Cell element, the *data* register (REG) stores the data and a Valid bit. On the right side of the figure, the adjacent Cell *write/read* pointer registers are depicted. When the *write* register of the Cell in the right (ptok_in = 1) is asserted and a write operation is performed (en_put = 1), the *data* register stores the input data (data_put). Likewise, when the *read* register of the Cell in the right (gtok_in = 1) is asserted and a read

operation is performed (en_get = 1), the *data* register (REG) is read through a tri-state buffer to the output data bus (data_get).

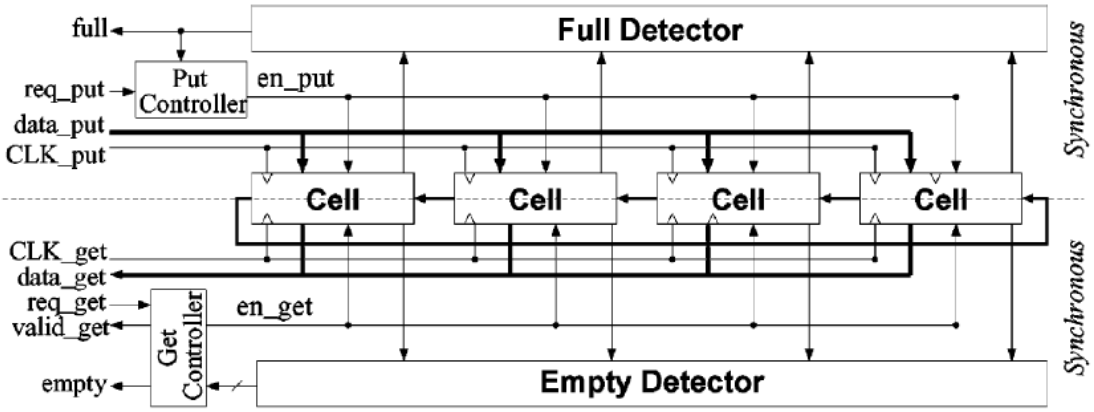


Figure A.3 TCSN FIFO overview [Chelcea04]

An asynchronous SR register per CELL element computes the state of the FIFO. If output *f_i* is asserted, the *data* register *i* contains data to be read. Likewise, if the output e_i is asserted, the *data* register *i* does not contain data to be read. This information is used on the detection of the full and empty conditions.

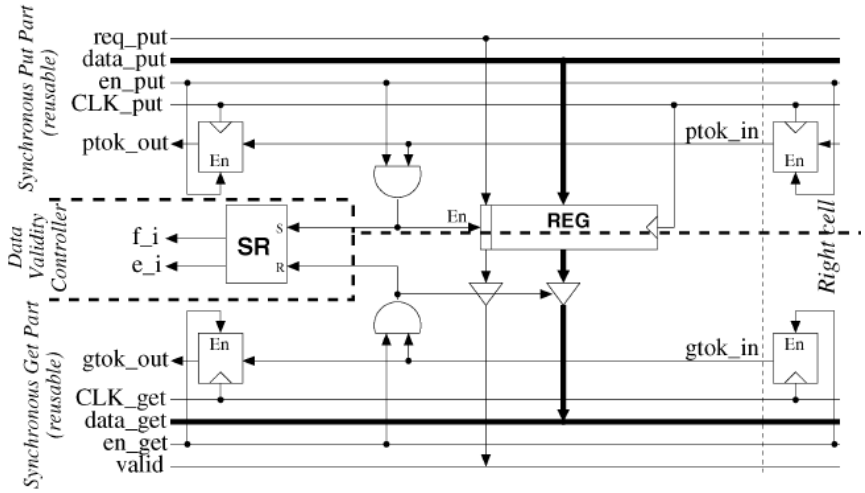


Figure A.4 TCSN FIFO: Cell element [Chelcea04]

The Full detector is composed by NMOS transistors, a PMOS transistor and a two-flop synchronizer as depicted in Figure A.5a. The detector analyze if at least two consecutive data registers are empty (example e_0=1 and e_1=1). If two of them are empty, the output is Low; otherwise, it is high. Finally, this signal is synchronized using a two-flop synchronizer to obtain the Full signal.

The Put controller is depicted in Figure A.5b. Its output is asserted when the producer system request a write operation (req_put =1) and the FIFO is not full. Its output is sent to all the Cell elements of the FIFO.

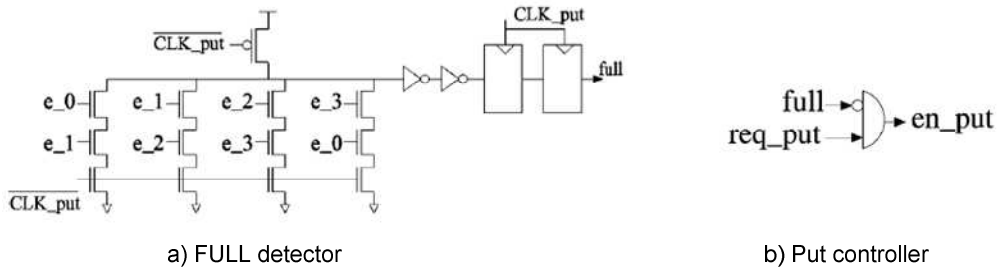


Figure A.5 TCSN FIFO: Full detector and put controller [Chelcea04]

The empty detector is composed by three modules: the *New* empty detector (Figure A.6a), the *Normal* empty detector (Figure A.6b), and the *Empty* detector (Figure A.7). The *New* empty detector computation is similar to the Full detector. If two successive cells are full (example f_0=1 and f_1=1), the *ne* signal is asserted after synchronization. The *ne* signal detects when the FIFO contains at least two elements. In order to detect at least one element in the FIFO, the *Normal* detector is needed.

Finally, the *ne* signal and the *oe* signal (from the *Normal* detector) are combined using a AND gate to generate the FIFO Empty signal (Figure A.7)

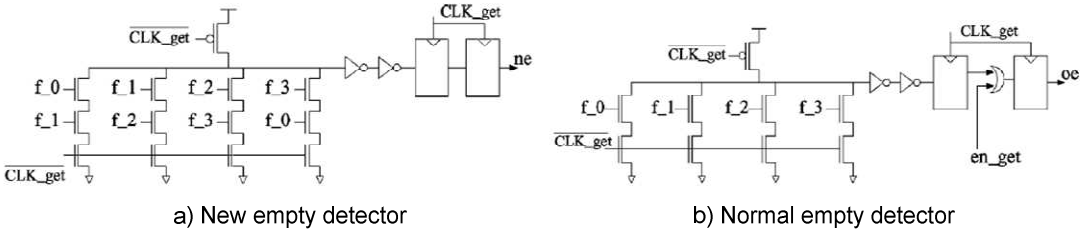


Figure A.6 TCSN FIFO: New and normal empty detectors [Chelcea04]

The Get controller generates the *en_get* signal, which is asserted when the consumer requires a FIFO read operation (req_get = 1) and the FIFO is not empty. Furthermore, a data valid signal (valid_get) is generated when the valid bit of the data register (Figure A.4) is asserted, the consumer required a FIFO read operation and the FIFO is not empty.

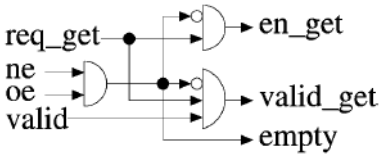


Figure A.7 TCSN FIFO: Get controller and empty detector [Chelcea04]

A.2.2 Analysis

The TCSN FIFO is a compact and well-optimized architecture. The one-hot encoding algorithm allows a distributed implementation of the architecture in a modular manner. However, the design has some limiting aspects:

- **Custom cells**: The architecture requires full custom cells to be implemented. The Full and Empty detectors requires special NMOS and PMOS transistor designs. These detectors could be designed in standard cells, despite of the increase of the area.
- **Clock relationship**: If the producer clock frequency is more than three times higher/lower than the consumer clock frequency, the FIFO can be useless. This problem comes from the asynchronous RS latch. When a write operation is performed, the *f_i* signal of the RS latch is asserted asynchronously even when no data is written yet. If the consumer frequency is more than three times higher than the producer one, the empty detector is deasserted. Under these circumstances, the consumer could try to read a new data that is not yet written. Thus, the FIFO operation is not correct. The same phenomenon occurs when the consumer starts to reads a FIFO data and the producer is three or more times faster than the consumer does. Under these circumstances, the producer writes a new data in the same data register while the consumer reads tries to read the old one.
- **Glitch**: If the producer system connected to the FIFO has glitches in the write request (req_put) signal, the asynchronous RS latch (Figure A.4) will toggle to Full state even when no data is written into the FIFO.
- Finally, the FIFO is protected by international patents [Chelcea07].

A.3 J. Jex et al. FIFO

The J. Jex at al. FIFO is a register-based FIFO with a programmable settling time synchronizer [Jex97][Dike00]. This programmability is used to improve the robustness of the synchronizer to the metastability. It was used on the Intel TeraFlops computer developed at Sandia Labs in 1994, which contains about 9200 Xeon type processors. Thus, there are 9200 interfaces between a 66 MHz bus and a self-timed circuit able to burst 200 MHz.

A.3.1 Architecture

Its architecture contains a Write pointer a Read pointer, a status register, an Empty detector, a Full detector, and data registers. The Write and Read pointer use one-hot encoding algorithm to identify the register to write and the register to read. Thus, the depth of the FIFO is equal to the number of bits of the write pointer. The write pointer bits controls the write_enable (Wen) of the data register while the read pointer bits controls the output_enable (Oen) input of the data registers as shown in Figure A.8.

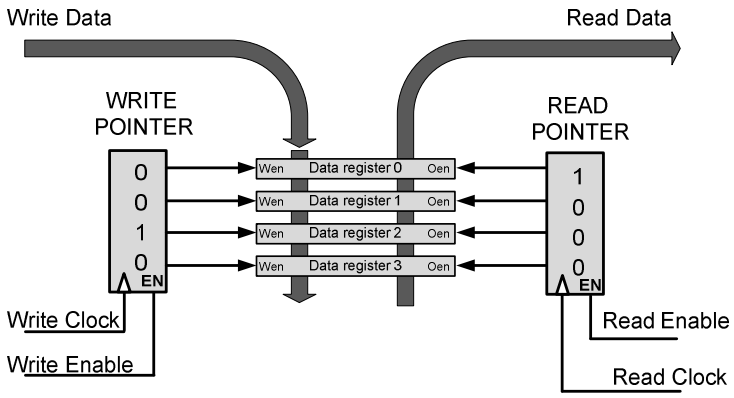


Figure A.8 J. Jex at al. FIFO data path

A status register is an N-bit register where N is the depth of the FIFO. The status register content identifies the data registers that contain valid data. If the bit *i* of the status register is asserted, the data register *i* contains valid data. Figure A.9 shows the status register, the Full detector, and the Empty detector.

The status register is controlled by the write pointer and the write_enable signal of the FIFO, as shown in Figure A.10. When a new data is written into the FIFO, the status register bit of the corresponding written data register is asserted. Then, the status register is synchronized to the read clock domain using a two-flop

synchronizer per bit. The synchronized signals are used to detect the empty condition on the Empty detector, which is described hereinafter.

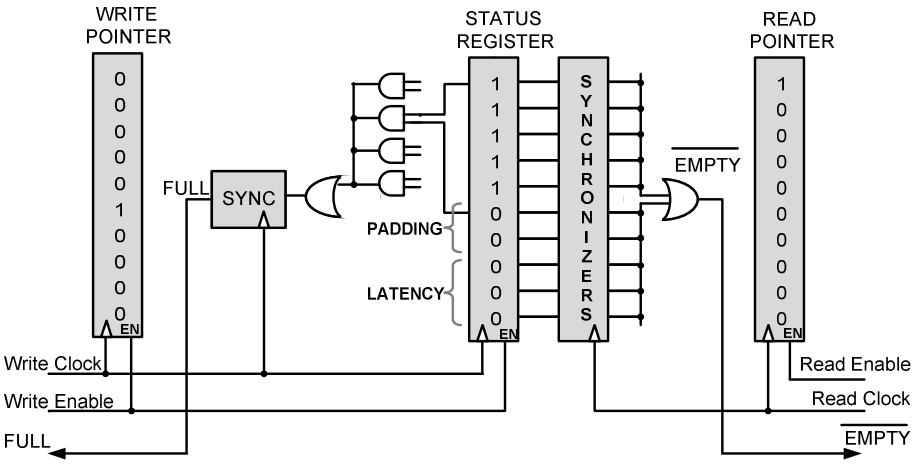


Figure A.9 Jex at al. FIFO: Full and Empty detectors [Dike00]

When the read side performs a read operation on data_register *i*, the status register *i*, and the two-flop synchronizer *i* are reset using the read_pointer *i* bit. Thus, the write side can perform a new write operation on data on register *i*.

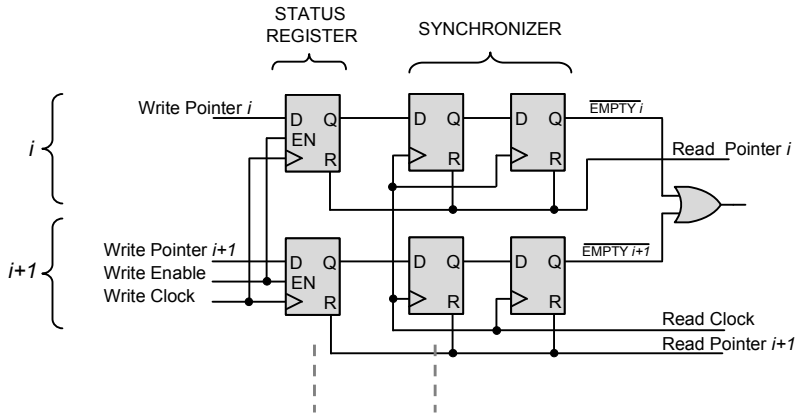


Figure A.10 Jex at al. FIFO: status register and synchronizers [Dike00]

The Empty detector is a simple OR gate collecting all the synchronized signals of the status register, as shown in Figure A.9. If there are one or more asserted bits on the synchronized status register the FIFO is not empty.

The Full detector compares with AND gates the pairs of bits of the status register to detect half-full FIFO conditions as shown in Figure A.9. Status register bit *i* is compared to *((N-1)/2+i) mod N* bit using a AND gate. Thus, if the result of the AND gate for any *i* is TRUE, the FIFO contains at least N/2 elements. All the AND gates are collected using an N-input OR gate and later synchronized to obtain the FULL signal.

Due to the synchronization latency of the synchronizers, the detection of the Full condition is anticipated.

In order to improve the robustness of the FIFO against the metastability failure, the synchronizers use a programmable settling time system. Thus, the settling time of the synchronizer can be modified from one clock cycle to four clock cycles. Therefore, the two-flop synchronized becomes a three-flop or four-flop synchronizer. However, the latency of the FIFO increases when the settling time increases.

Detailed description of the architecture can be found on patent [Jex97] and in tutorial [Dike00].

A.3.2 Analysis

All the metastability issues are correctly analyzed. The synchronizers are well dimensioned and the settling time can be increased in order to improve the metastability robustness. The design is simple and can be implemented using standard cells. However, no mesochronous adaptation is proposed.

The throughput of the FIFO can be maximized to 100% for large FIFOs; however, the half Full FIFO detection and the latency of the synchronizers reduce the FIFO throughput when the FIFO depth is small. The minimum FIFO depth to archive 100% throughput is 14 words. The half-full detection of the Full detector, limits the performance of the FIFO because it is nearly impossible to full all the FIFO elements. Thus, the useful depth of the FIFO is N/2+4 (using the smaller synchronization latency), where 4 is the synchronization latency of Full and Empty.

Finally, the FIFO architecture is protected by Intel Corporation on US Patent [Jex97].

Appendix B

Integration of the DSPIN Network into the FAUST Platform

In this appendix, we analyze the problems that must be solved to replace the ANOC NoC by the DSPIN NoC in the FAUST stream-oriented platform. We analyze how the DSPIN network-on-chip, originally designed for shared memory multi-processors architectures, can support stream-oriented architectures.

In section B.1, the ANOC and DSPIN architectures are compared in terms of routing algorithm, switching algorithm, packet definition, and clocking techniques. In section B.2, the IP integration template for ANOC is depicted and adapted to the DSPIN network. In section B.3, we describe how the FAUST topology has been rearranged in order to support the X-first routing algorithm. Finally, we describe in section B.4, how the DSPIN FIFOs have been dimensioned to support the real-time constraints of the FAUST application.

B.1 Architecture Comparison

DSPIN and ANOC are similar NoC architecture. However, DSPIN uses synchronous cells while ANOC uses asynchronous cells. Both architectures offer the same type of traffic and they can be exchanged. Towards that end, their similarities and differences are analyzed in the next table:

Table B.1 ANOC and DSPIN architecture comparison

	ANOC	DSPIN
Topology	Irregular	Regular 2D mesh
Router arity	5 port router	5 port router
Routing technique	Source routing	Address-based X-First algorithm
Switching technique	Wormhole	Wormhole
Flit size	34 bits	34 bits (generic)
Flit payload	32 bits	32 bits (generic)
Flow control bits	Begin of packet (BOP) End of packet (EOP)	Begin of packet (BOP) End of packet (EOP)
Routing overhead and capability	18-bits, allowing 9 routing hops Path extension is possible	8-bits, allowing any architecture up to 16*16 clusters
Programming model	Message passing	Shared-memory (2 routers per cluster) Message-passing (1 router per cluster)
Clocking scheme	Fully asynchronous (QDI) with GALS interfaces	Multi-synchronous with mesochronous interfaces
Metastability issues	Metastable-free inside routers (4 phase protocol), GALS FIFO interfaces on local ports	Resolved by bi-synchronous FIFOs
Virtual channels	Best effort and Guaranteed service	Best effort and Guaranteed service
Flow control protocol	Send/accept asynchronous handshake	FIFO protocol (Write and WriteOk)
Clock tree	None	One per router
Physical implementation	Hard macro	Soft macro distributed on five modules
Long wires	Inter-router wires	Intra-cluster wires

DSPIN and ANOC use similar packet format. Figure B.1 shows the ANOC and DSPIN packet definition. DSPIN having a generic flit size, its length can be set as long as the ANOC flit size, 34 bits. Thus, both architectures have 32 bits of packet payload. ANOC uses 18 bits on the first flit for the source routing information while DSPIN uses 8 bits for the destination address. Moreover, both architectures use the same control bits *Begin_of_Packet* (BOP) and *End_of_Packet* (EOP).

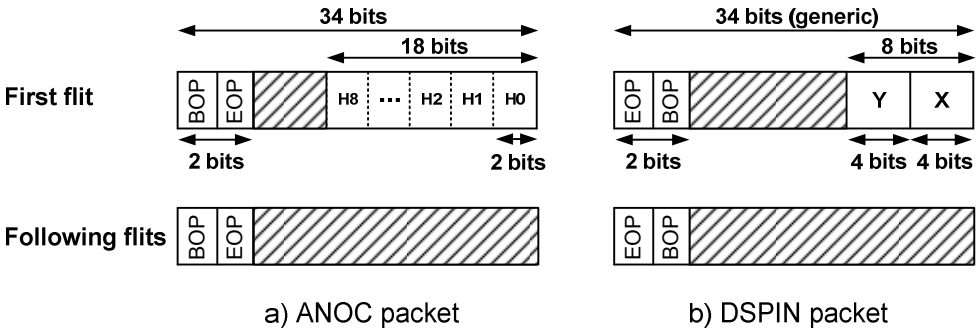


Figure B.1 ANOC and DSPIN packet definition

B.2 Protocol Conversion

This section presents the *protocol_conversion* module, which replaces the *GALS_interface* module as shown in Figure B.2.

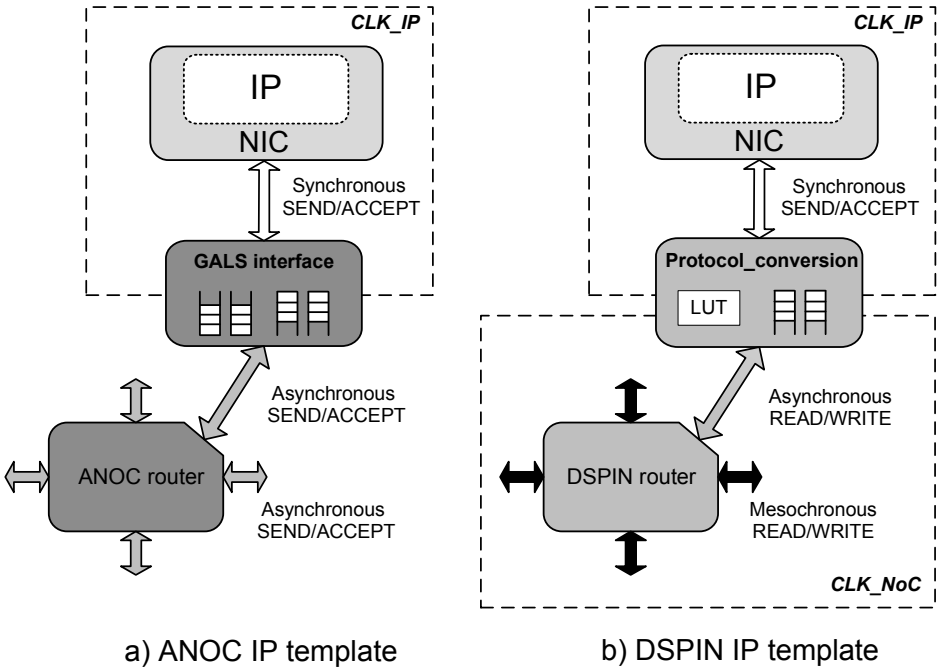


Figure B.2 IP integration detail

For ANOC, the *GALS_interface* module performs the adaptation of the asynchronous send/accept protocol to the synchronous send/accept protocol of the local subsystem. Moreover, this module contains the local input and output FIFOs, which performs the synchronization of the asynchronous-data to the local clock frequency.

For DSPIN, the *protocol_conversion* module is directly connected to local subsystem through the synchronous Send/Accept interface. Moreover, this module

contains the synchronization FIFOs to the local clock frequency. The implementation of this module is detailed hereinafter.

B.2.1 Flow Control Conversion

All FAUST local subsystems use the synchronous send/accept protocol. This flow control protocol works as follows:

- **Send**: The producer informs that a data is valid to be sent on the *current* cycle.
- **Accept**: The consumer is ready to accept one data at the *next* cycle.

The producer is allowed to transfer the data, if only if, at the previous cycle the consumer asserted the *Accept* signal. Therefore, the transfer of the data depends on the state of the *Accept* signal on the *previous* cycle.

DSPIN uses a FIFO protocol controlled by two signals:

- **Write**: The producer informs that a data is valid to be sent on the *current* cycle.
- **Read**: The consumer is ready to accept one data on the *current* cycle.

The producer is allowed to transfer the data, if only if, the consumer is ready to accept a data on the *current* cycle. Data transfers depend only on the actual state of the control signals.

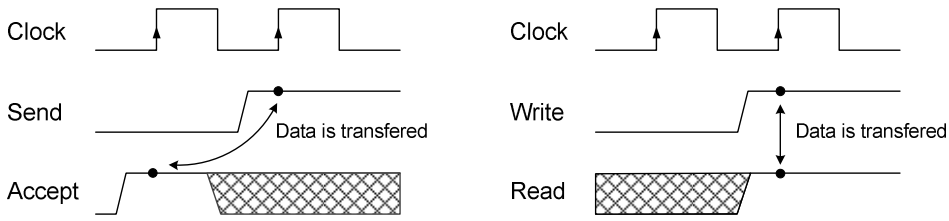


Figure B.3 Send/Accept and Write/Read protocol

The conversion module translates the Send/Accept flow control signal into the DSPIN flow control signals following these conversions:

- If the producer uses the write/read protocol, the conversion to the send/accept protocol is simple (Figure B.4a). A register delays the *Accept* signal one clock cycle to generate the *Read* signal of the FIFO. Moreover, a AND gate asserts the *Send* signal, if only if, the FIFO is ready to accept need data (ROK = 1) and the *Accept* signal was asserted on the previous clock cycle.
- When the producer uses the Send/Accept protocol, the FIFO has to be internally modified (Figure B.4b). The *Accept* signal is the inverted value

of the intermediate signal of the FULL synchronizer. Hence, the *Accept* signal is advanced of one clock cycle. Moreover, the internal *Write* signal of the FIFO is asserted, if only if, the *Accept* has been asserted on the previous clock cycle and the *Send* signal asserted.

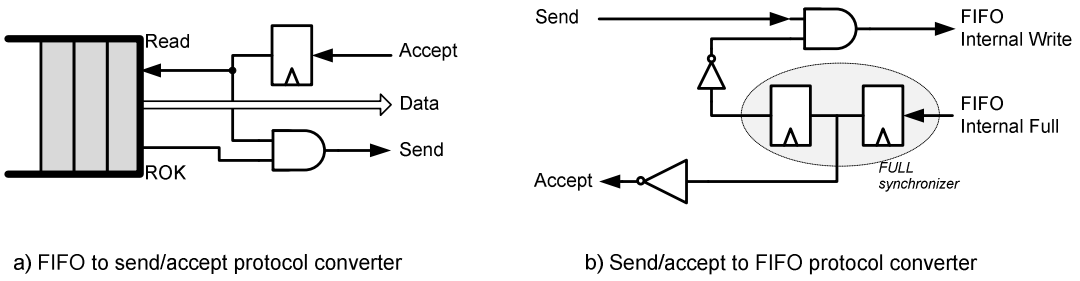


Figure B.4 Flow control signal converters between send/accept and FIFO protocol

B.2.2 Packet Address Conversion

DSPIN and ANOC use a similar packet format. However, ANOC uses a source routing algorithm and DSPIN uses an address-based routing algorithm. Thus, it is necessary to convert all the source routing bits into address-based bits. In order to avoid any modification on the application software, the routing algorithm conversion is performed by the hardware using a Look-Up Table (LUT). Therefore, each *protocol_conversion* module contains a hard-wired LUT. The LUT converts the 18-bits of source routing to the 8-bits of destination address (Figure 2.16). As the routing information is just inserted on the first flit of the packet, the *protocol_conversion* module replaces this data just when the *begin_of_packet* signal is asserted. Otherwise, the packet data is not modified. This solution is not optimized, as it uses a hard-wired LUT but we did not want to modify the application software for this experiment.

The content of the LUT have been generated analyzing all the paths used by the selected application on the FAUST platform. A *spy* module inserted between the *GALS_interface* module and the NI of each FAUST module (Figure B.5) has been used to extract all the communications of the platform.

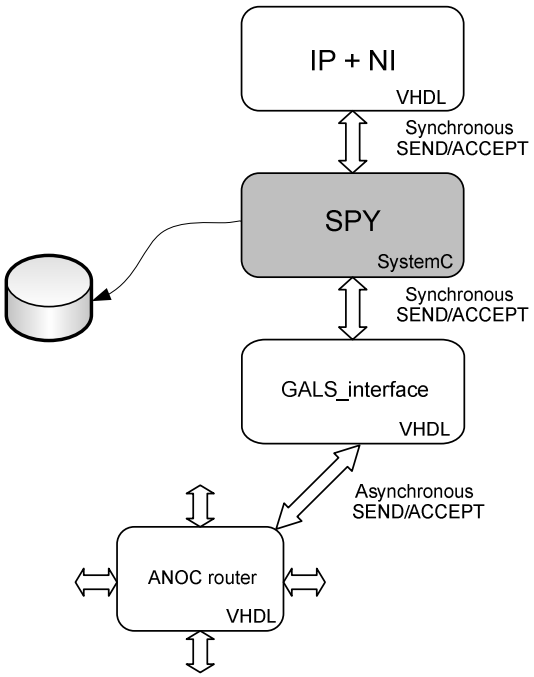


Figure B.5 SPY module

Table B.2 shows the information obtained by the *spy* module. Source and destination denotes the producer and consumer of the packet respectively. The *np_1* module, which is connected to top input external port of the FAUST chip, has many different packet destinations. This phenomenon is the result of the initialization of the FAUST chip, where all the modules receive their configuration by the external port (see Figure 2.17 for the FAUST architecture). Accordingly, the *np_1* module has to forward this configuration to all the modules of the system.

As explained in previous paragraph, each IP requires a dedicated *packet_conversion* module as its pair source-destination LUT is different. However, in order to reduce the number of different LUTs, we developed three different LUTs covering all the previous table cases. Thus, the same LUT is reused for various *packet_conversion* modules.

Table B.2 Routing information of FAUST modules

Source	Destination	Routing path	Number of packets	Channel
dec	rx_bit	ESL	4608	VC0
dec	np_1	NNWWNL	832	VC0
dmap	rx_bit	EL	4416	VC0
dmap	RAM_ext	NNEL	17664	VC0
equal	rx_ofdm	SL	17664	VC0
equal	est	EL	17664	VC0
equal	rx_fht	SEL	17664	VC0
est	RAM_ext	EENL	4416	VC0
est	equal	WL	17664	VC0
est	rx_ofdm	WSL	2944	VC0
np_1	drh	EENNNNNNN	1	VC0
np_1	syn	SSSWL	14	VC0
np_1	RAM1	SL	35436	VC0
np_1	RAM Ext	SEEEL	8	VC0
np_1	rot	SSWL	16	VC0
np_1	equal	SSL	12	VC0
np_1	est	SSEL	11	VC0
np_1	rx_fht	SSSEL	9	VC0
np_1	rx_ofdm	SSSL	17	VC0
np_1	rx_bit	SSSEEEL	9	VC0
np_1	dmap	SSSEEL	267	VC0
np_1	dec	SSEEL	8	VC0
np_1	dec	EESSL	810	VC0
RAM_ext	dmap	WSSL	23552	VC0
RAM_ext	est	WWSL	549	VC0
RAM_ext	rx_fht	WWSSL	8583	VC0
RAM1	rot	WSL	28672	VC0
RAM1	syn	WSSL	801	VC0
RAM1	np_1	NNNNNNNNN	35004	VC0
rot	ram1	NEL	28665	VC0
rot	rx_ofdm	SEL	28672	VC0
rx_bit	dec	NWL	4608	VC0
rx_bit	dmap	WL	4416	VC0
rx_fht	RAM_ext	NNEEL	8832	VC0
rx_fht	equal	NWL	17664	VC0
rx_ofdm	rot	WNL	28672	VC0
rx_ofdm	est	NEL	2944	VC0
rx_ofdm	equal	NL	17664	VC0
syn	np_1	ENNNNWNNN	8	VC0
syn	RAM1	NNEL	801	VC0

B.3 Topology Rearrangement

DSPIN NoC is designed for regular 2D mesh topologies. However, the FAUST architecture has some irregularities that are easily handled by the source routing algorithm of ANOC, but have to be arranged for DSPIN.

Firstly, the local subsystem on DSPIN NoC has to be connected to the local port of the DSPIN router. Otherwise, the X-first algorithm cannot route properly the packets over the network. In the FAUST architecture, the modules *np_1 (top input port)*, *ahb* (clk & test ctrl), *enc* (turbo coder), *dec* (convolution coder), and *exp* are connected to non-local ports. Fortunately, the *enc*, *dec*, and both *exp* modules are not used by the selected software application. Consequently, the topology of FAUST was rearranged to connect all the used modules on the local ports of the DSPIN routers (see Figure 2.17). The new topology is depicted in Figure B.6 where the orange lines show the modified connections. The modules depicted in green are physically implemented but not used by the application.

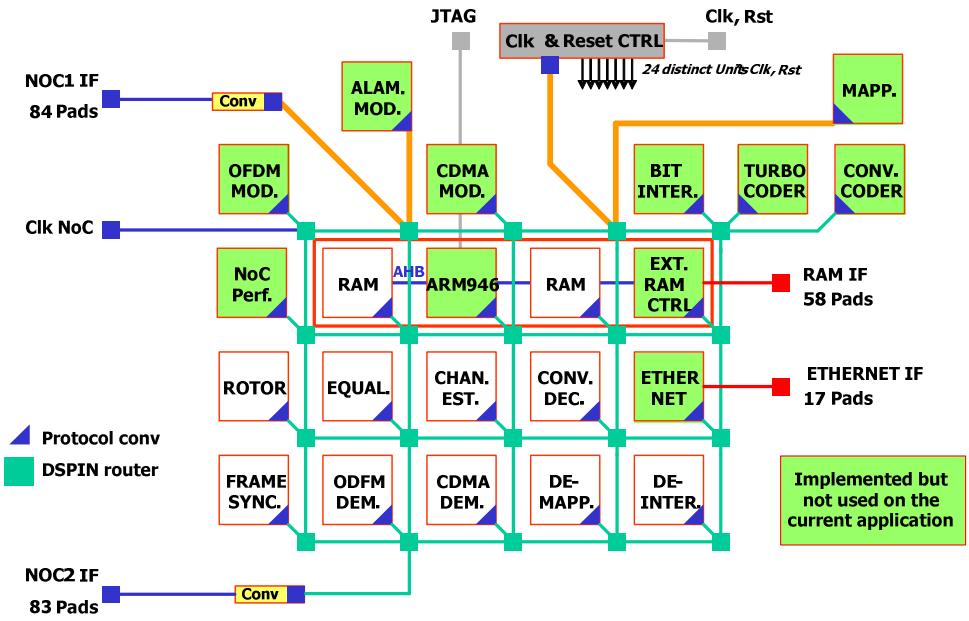


Figure B.6 Rearanged totpology of FAUST chip for DSPIN

B.4 FIFO Dimensioning

The analysis of the *spy* module results reveals some path conflicts when the X-first routing algorithm is used. These path conflicts results in multiple utilization of the same GS links by two or more producer-consumer pairs. Consequently, the total throughput of the link is shared between all the producer-consumer pairs. As the ANOC uses the source routing algorithm, these path conflicts are resolved by rerouting the conflicting path to other links. Consequently, some links of the FAUST platform are more congested with DSPIN than with ANOC. Table B.3 shows the three routing conflicts when the X-first algorithm is used.

Table B.3 Routing conflicts using the X-first algorithm

Number of packets	Source	Destination	Routing path
17664	dmap	RAM_ext	NNEL
8832	rx_fht	to RAM_ext	NNEEL
28672	rot	rx_ofdm	SEL
17664	equal	rx_ofdm	SL
17664	rx_fht	equal	NWL
17664	rx_ofdm	equal	NL

Two solutions have been considered to minimize these conflicts.

- Rearrange the mapping of the subsystem to avoid the congestion. This solution has been abandoned as the modification on the mapping has an impact on the floorplanning of the chip and many memory hard macros are used. Consequently, the placement has not been modified to minimize floorplanning differences between the implementation of ANOC and DSPIN.
- Increase the depth of the DSPIN FIFOs to support the conflicting traffics without reducing the performance of the system. This solution had been chosen as it does not modify the chip topology and it is compatible with the DSPIN architecture.

The FIFO depth of the DSPIN router has been dimensioned by simulating the whole system with different DSPIN FIFO depths. The FAUST application demodulates the OFDM signal with a maximum processing time per OFDM frame of 600us. Consequently, the dimensioning of the FIFO has to guarantee this processing

time for the worst-case condition. Hence, the DSPIN clock frequency is set as low as the worst-case condition of the system clock frequency (150 MHz). Table B.4 shows the processing time of one OFDM frame in function of the FIFO depth at 150MHz.

Table B.4 Processing time of one OFDM frame in function of DSPIN FIFO depth at 150MHz

FIFO depth (words)	Processing time per OFDM frame
5	773 μs
6	577 μs
7	577 μs

The minimum GS FIFO depth to guarantee the system performance is six words. Beyond this depth, the FIFO depth does not improve the system performances. Finally, we have chosen to implement a GS FIFOs depth of seven words because the best effort and guaranteed service channels are equilibrated with seven words each one.

Appendix C

Power Consumption Estimation in the FAUST platform

This appendix details the power consumption estimation for the DSPIN and ANOC implementations in the FAUST platform. The power estimation is performed using back-annotated simulations of the implemented gate-level netlist. The simulation application is a full OFDM demodulation frame. The comparison of the ANOC and DSPIN power consumption obtained in this appendix is analyzed in Chapter 5.

The first section presents the power consumption estimation methodology employed. The power consumption estimation for DSPIN and ANOC is detailed in sections C.2 and C.3.

C.1 Power Consumption Estimation Methodology

The PrimPower tool was used to estimate the power consumption for both architectures DSPIN and ANOC. The input files required are the gate-level netlist, the *sdf* (parasites) file, and the *vcd* (stimuli) file. The parasites file was extracted from the physical layout on typical conditions. The stimuli file, is the back-annotation simulation of the gate-level netlist. This stimuli file contains all the transitions states of the signals on the selected modules; thus, even the glitch transition can be computed.

The tool computes the detailed power consumption module by module. The switching, internal, and glitch power is reported. Moreover, it is possible to visualize the power consumption over the time of each individual module. An example of this visualization is depicted in Figure C.1.

C.2 DSPIN Power Consumption Estimation

The DSPIN routers, NIC FIFOs and clock-trees power consumption were extracted for two implementations; with and without clock-gating.

C.2.1 Without Clock-Gating

The power consumption of the non clock-gating version of DSPIN was simulated for 274MHz and 149MHz. The power consumption over the time showed that the power consumption of the DSPIN router do not vary much in function of the router activity. At 149MHz, a full activity router consumes 10.3mW while a non-active router consumes 6.6mW. Moreover, the registers power consumption is 86% of the total DSPIN router power at 274MHz. The leakage power consumption is very low 0.5-0.7% of the total power, as the standard cells used were Low Power cells with high Vt transistors. Table C.1 details the average power consumption of a DSPIN router.

Table C.1 Power consomption of DSPIN router without clock-gating

	149 MHz	274 MHz
Power consumption per router	~ 9 mW	~ 14 mW
Leakage power	~ 0.7 %	~ 0.5 %
Dynamic power	~ 99.3 %	~ 99.5 %
- Switching power	~ 6.0 %	~ 3.0 %
- Internal power	~ 94.0 %	~ 97.0 %

Table C.2 shows the power consumption of each DSPIN router at 149 MHz. The position in the table corresponds to the position of the router on the network topology (see Figure 2.17). The gray cells indicate the non-active routers, because they are not used by the software application. The aggregated power consumption of the DSPIN routers is 165 mW.

Table C.2 Power consomption of DSPIN routers without clock-gating at 149 MHz

6.62 mW	7.97 mW	6.58 mW	6.64 mW	4.03 mW
7.31 mW	10.30 mW	8.41 mW	8.74 mW	9.01 mW
9.18 mW	10.39 mW	9.24 mW	9.49 mW	7.30 mW
8.47 mW	9.67 mW	8.80 mW	9.06 mW	8.62 mW

Table C.3 shows the DSPIN clock-tree power consumption for two clock frequencies. The clock tree computed here includes the DSPIN router and the DSPIN NIC FIFOs clock trees. The *Top clock tree* corresponds to the mesochronous clock tree, while the *Bottom clock tree* corresponds to the synchronous DSPIN route clock tree.

Table C.3 DSPIN clock-tree power consumption without clock-gating

	Power consumption at 149 MHz	**Power consumption at 274 MHz**
Top clock tree	1.26 mW	2.31 mW
Bottom clock tree	151.00 mW	274.00 mW
Total	152.26 mW	280.31 mW

99.9 % of the clock-tree power consumption is due to the dynamic power consumption, where 75% is switching power and 25% is internal power. Each DSPIN clock tree power (a branch of the bottom clock-tree) consumes 13.2 mW at 274 MHz and 7.3 mW at 149 MHz. Consequently, the power consumption of the DSPIN clock-tree consumes as much as the DSPIN router itself. Moreover, the total power consumption routers and clock-trees are 317 mW at 149 MHz.

C.2.2 With Clock-Gating

With the clock-gating technique, the power consumption was reduced. Table C.4 and Table C.5 show the power consumption of the DSPIN routers at 149 MHz and 289MHz. The position in the table corresponds to the position of the router on the network topology (see Figure 2.17). The gray cells indicate the non-active routers, because they are not used by the software application. The DSPIN routers power consumption is 54.25 mW at 149MHz and 92.17 mW at 289MHz. With the introduction of the clock-gating, the power consumption of the DSPIN routers is reduced by 67% (at 149MHz).

With the clock gating mechanism, the increase of the clock frequency does no longer force a linear increase of the power consumption. Otherwise, the power consumption at 289MHz would be 105 mW instead of 92.17 mW, thus saving 12%.

Table C.4 Power consomption of DSPIN routers with clock-gating at 149MHz

2.02 mW	3.05 mW	2.06 mW	2.06 mW	1.63 mW
2.75 mW	4.23 mW	2.32 mW	2.51 mW	2.64 mW
3.95 mW	3.96 mW	3.00 mW	2.52 mW	2.13 mW
2.45 mW	3.27 mW	2.61 mW	2.61 mW	2.43 mW

Table C.5 Power consomption of DSPIN routers with clock-gating at 289MHz

3.92 mW	4.76 mW	3.93 mW	3.72 mW	3.09 mW
4.73 mW	6.40 mW	4.17 mW	4.48 mW	4.53 mW
5.95 mW	6.09 mW	4.93 mW	4.49 mW	4.00 mW
4.33 mW	5.27 mW	4.55 mW	4.53 mW	4.30 mW

Figure C.1 shows a detailed analysis of the power consumption on router (1,2) at 149MHz. The first row shows the aggregated power consumption of the router while the other rows show the detailed power consumption of the DSPIN router modules. cInEast is the sender submodule of East DSPIN module while cOutEast is the receiver submodule of East DSPIN module. Submodule cInEast does not have activity, thus its power consumption is very low. Submodule cInSouth receive a long packet in the middle of the scope while cInLocal receive many long packets. It is possible to identify the power consumption due to packet transmission and the power reduction when no packet is sent. Submodule cWires contains the buffers interconnecting the modules, the long wires of DSPIN router.

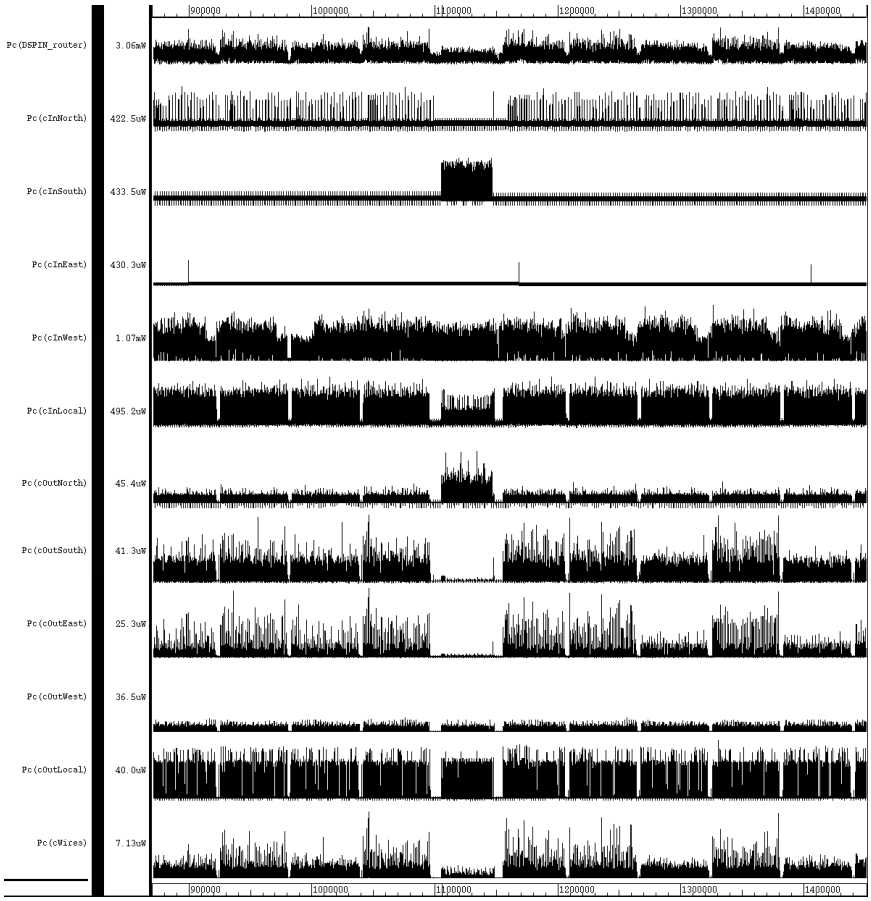


Figure C.1 Power analysis of router (1,2) at 149MHz

The power consumption of the two FIFOs contained in the NIC of was extracted for 149MHz and 289MHz. In this physical implementation, these FIFOs are located in the *protocol_conversion* module; see Appendix B for further details. Table C.6 shows the power consumption for each pair of FIFOs on the topology.

Table C.6 Power consumption of NIC FIFOs at 149MHz (at 289MHz)

242.8 μW (462 μW)	478.9 μW (697 μW)	245.2 μW (463 μW)	481.9 μW (698 μW)	246.0 μW (465 μW)
246.5 μW (465 μW)	919.9 μW (1319 μW)	500.2 μW (718 μW)	501.7 μW (722 μW)	559.4 μW (792 μW)
842.4 μW (1110 μW)	642.4 μW (937 μW)	519.9 μW (742 μW)	487.6 μW (713 μW)	245.4 μW (464 μW)
447.2 μW (665 μW)	834.6 μW (1118 μW)	593.1 μW (842 μW)	597.4 μW (850 μW)	464.0 μW (689 μW)

Table C.7 shows the clock-tree power consumption for two clock frequencies. Compared to the non clock-gating implementation, the power consumption is reduced by 67 %.

Table C.7 DSPIN clock-tree power consumption with clock-gating

	Power consumption at 149 MHz	Power consumption at 289 MHz
Top clock tree	1.23 mW	2.39 mW
Bottom clock tree	47.70 mW	92.30 mW
Total	48.93 mW	94.69 mW

Table C.8 shows the total power consumption with clock-gating for 149 MHz and 289 MHz. Around 50% of the total power consumed is consumed by the clock-tree. Comparing with the non clock-gating implementation, this implementation has saved 67% of power consumption at 149 MHz.

Table C.8 Total power consumption with clock-gating

	Power consumption at 149 MHz	Power consumption at 289 MHz
DSPIN routers	54.25 mW	92.17 mW
FIFOs on NIC	9.96 mW	14.92 mW
Clock tree	48.93 mW	94.69 mW
Total (all routers)	113.14 mW	201.78 mW
Total (per router)	5.65 mW	10.08 mW

C.3 ANOC Power Consumption Estimation

The ANOC power consumption estimation has been performed using the same methodology as the DSPIN one, and the same OFDM demodulation application has been used. The power consumption have been analyzed for the 15 bottom ANOC routers rather than for the 20 because these the other 5 do not interact with the selected application. Table C.9 shows the leakage and dynamic power consumption of the 15 bottom ANOC routers (see Figure 2.17). The total power consumed by these routers is 31.04 mW while the leakage power is around 0.37 μW per router.

Table C.9 Power consomption of ANOC routers

?.? mW	?.? mW	?.? mW	?.? mW	?.? mW
1.93 mW	3.75 mW	0.83 mW	1.85 mW	1.85 mW
3.40 mW	3.30 mW	2.04 mW	1.65 mW	0.66 mW
1.89 mW	3.76 mW	1.74 mW	1.55 mW	0.84 mW

The power consumption of the *GALS_interface* module was computed for the 15 bottom clusters as shown in Table C.10. The leakage power consumption is around 0.24 μW for each module.

Table C.10 Power consomption of GALS_interface modules

?.? mW	?.? mW	?.? mW	?.? mW	?.? mW
0.024 mW	3.646 mW	0.026 mW	0.026 mW	2.096 mW
3.166 mW	3.185 mW	1.797 mW	1.454 mW	0.026 mW
0.026 mW	3.076 mW	2.265 mW	2.034 mW	1.556 mW

Abbreviations

ANOC	Asynchronous Network-on-Chip
ASIC	Application Specific Integrated Circuit
BE	Best Effort
BIST	Built in Self-Test
CMOS	Complementary Metal Oxide Semiconductor
DSPIN	Distributed Scalable Programmable Integrated Network
EDA	Electronic design automation
FAUST	Flexible Architecture of Unified Systems for Telecom
FIFO	First In, First Out
GALS	Globally Asynchronous, Locally Synchronous
GDSII	Graphic Data System II
GS	Guaranteed Service
MANGO	Message-passing Asynchronous Network-on-Chip providing Guaranteed services through OCP interfaces
MPEG	Moving Picture Experts Group
NIC	Network Interface Controller
NMOS	N-channel Metal Oxide Semiconductor
NoC	Network-on-Chip
OFDM	Orthogonal Frequency-Division Multiplexing
PLL	Phase Locked Loop
PMOS	P-channel Metal Oxide Semiconductor
QDI	Quasi Delay Insensitive
QNoC	Quality of Service Network-on-Chip
RAM	Random Access Memory
RC	Resistance Capacitance
RTL	Register Transfer Level
SDF	Standard Delay Format
SLID	Synchronous Latency Insensitive Designs
SoC	System-on-Chip
SPIN	Scalable Programmable Integrated Network

SRAM	Static Random Access Memory
TDM	Time Division Multiplexing
VC	Virtual Channel
VCI	Virtual Component Interface
VHDL	VHSIC Hardware Description Language
VHSIC	Very High Speed Integrated Circuits
VLSI	Very Large Scale Integration

Bibliography

[ALLIAN] http://asim.lip6/recherche/alliance

[Andria03] A. Andriahantenaina and A. Greiner "Micro-network for SoC: Implementation of a 32-port SPIN network," Design Automation and Test in Europe (DATE 2003) pp. 1128-1129, March 2003.

[Andria06] A. Andriahantenaina, "Physical implementation of a 32-port SPIN micro-network (Implémentation matérielle d'un micro-réseau SPIN à 32 ports)," PhD thesis, The University of Pierre et Marie Curie, France, Jan. 2006.

[Apper07] R. Apperson, Z. Yu, M. Meeuwsen, T. Mohsenin, and B. Baas, "A Scalable Dual-Clock FIFO for Data Transfers between Arbitrary and Haltable Clock Domains," IEEE Trans. of Very Large Scale Integration Systems (TVLSI), vol. 15, no. 10, pp. 1125-1134, October 2007.

[ARM] www.arm.com

[Bartels06] C. Bartels, J. Huisken, K. Goossens, P. Groeneveld, and J. Meerbergen, "Comparison of An Æthereal Network on Chip and A Traditional Interconnect for A Multi-Processor DVB-T System on Chip,"in Proc. IFIP Int'l Conference on Very Large Scale Integration (VLSI-SoC), October 2006.

[Baut07] J. Bautista, "Tera-scale Computing - the Role of interconnects in Volume Compute platforms," IEEE Int. Interconnect Technology Conference, pp. 187-189, June 2007.

[Beigne05] E. Beigne, F. Clermidy, P. Vivet, A. Clouard, and M. Renaudin, "An Asynchronous NoC Architecture Providing Low Latency Service and its Multi-Level Design Framework," Proc. 11th Int. Symp. on Advanced Research in Asynchronous Circuits and Systems, ASYNC'2005, New-York, USA, pp. 54-63, March 2005.

[Beigne06] E. Beigne and P. Vivet, "Design of On-chip and Off-chip interfaces for a GALS NoC architecture," in Proc. 12th IEEE Int. Symp. on Asynchronous Circuits and Systems (ASYNC'06), pp. 172-181, March 2006.

[Benini02] L. Benini and G. De Micheli, "Networks on chip: A new SoC paradigm". IEEE Computer, vol. 35, issue 1, pp. 70-78, January 2002.

[Berens05] F. Berens et al., "Designing a multiple antenna MC-CDMA SoC for beyond 3G," Embedded Systems, San Francisco, USA, March 2005.

[Bertoz04] D. Bertozzi and L. Benini "Xpipes: A Network-on-Chip architecture for gigascale Systems-on-Chip," IEEE Circuits and Systems Magazine, Q2 2004.

[Bjerre04] T. Bjerregaard and J. Sparsø, "Virtual channel designs for guaranteeing bandwidth in asynchronous network-on-chip," In Proceedings of the IEEE Norchip Conference, 2004.

[Bjerre05a] T. Bjerregaard and J. Sparsø "A router architecture for connection-oriented service guarantees in the MANGO clockless Network-on-Chip," IEEE Proc. Design Automation and Test in Europe (DATE'05), March 2005.

[Bjerre05b] T. Bjerregaard and J. Sparsø, "A Scheduling Discipline for Latency and Bandwidth Guarantees in Asynchronous Network-on-chip," In Proc. Int. Symp. Asynchronous Circuits and Systems, pages 34-43. IEEE Computer Society Press, 2005.

[Bjerre05c] T. Bjerregaard, "The MANGO clockless network-on-chip: Concepts and implementation, " Ph.D. thesis, 2005.

[Bolotin03] E. Bolotin, I. Cidon, R. Ginosar, A. Kolodny, "Cost considerations in Network on Chip," Special issue on Networks on Chip, Integration - The VLSI Journal, 2003.

[Bolotin04] E. Bolotin, I. Cidon, R. Ginosar and A. Kolodny "QNoC: QoS architecture and design process for network on chip," Journal of Systems Architecture, 50(2-3), pp. 105-128, February 2004.

[Bolotin07] E. Bolotin, I. Cidon, R. Ginosar, A. Kolodny, "Routing Table Minimization for Irregular Mesh NoCs," Proc. Design, Automation, and Test in Europe (DATE'07), 2007.

[Bonon06] L. Bononi and N. Concer "Simulation and Analysis of Network on Chip Architectures: Ring, Spidergon and 2D Mesh," Proc. Design, Automation, and Test in Europe (DATE'06), pp. 154-159, 2006.

[Bot04] E. Botolin, I Cidon, R. Ginosaur, and A. Kolodny, "QNoC: QoS architecture and design process for Network on Chip," Journal of System Architecture, vol. 50, No. 2-3, February 2004.

[Burle05] W. Burleson, "On-Chip Interconnects: Circuits and Signaling from an MPSoC Perspective," International Forum on Application-Specific Multi-Processor SoC (MPSoC'05), July 2005.

[Caputa06] P. Caputa, and C. Svensson, "An on-chip delay- and skew-insensitive multicycle communication scheme," in IEEE International Solid-State Circuits Conference (ISSCC 2006), pp. 1765- 1774, Feb. 6-9, 2006.

[Chakra02] A. Chakraborty and M. R. Greenstreet, "A minimal source-synchronous interface," Proc. 15th IEEE ASIC/SOC Conference, pp. 443-447, Sept.2002.

[Chakra03] A. Chakraborty and M. R. Greenstreet, "Efficient self-timed interfaces for crossing clock domains," in Proc. 9th IEEE Int. Symp. Asynchronous Circuits Systems (ASYNC'03), pp. 78-88, 2003.

[Chapiro84] D. M. Chapiro, "Globally-Asynchronous Locally-Synchronous systems," PhD thesis, Stanford University, 1984.

[Chattop05] A. Chattopadhyay and Z. Zilic, "GALDS: A Complete Framework for Designing Multiclock ASICs and SoCs,' IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 13, no. 6, June 2005.

[Chelcea04] T. Chelcea and S. M. Nowick, "Robust interfaces for mixed-timing systems," IEEE Trans. on Very Large Scale Integration (VLSI) Systems, vol. 12, no. 8, pp. 857-873, August 2004.

[Chelcea07] T. Chelcea and S. M. Nowick, "Low latency circuits for mixed asynchronous and synchronous systems", Patents DE60130039D, US2004125665, US2004128413, US2002167337, WO0182053, 2007.

[Chen99] R. Y. Chen, N. Vijaykrishnan, and M. J. Irwin, "Clock power issues in system-on-a-chip designs," in Proc. IEEE Workshop on VLSI, pp. 48–53, 1999.

[Cler05] F. Clermidy, D. Varreau, and D. Lattard, "A NoC-based communication framework for seamless IP integration in complex systems," Proceedings of Design and Reuse IP-SOC'2005, Grenoble, France, pp. 279-283, Dec 2005.

[Coppo04] M. Coppola et al., "Spidergon: a novel on chip communication network," Proc. Int'l Symposium on System on Chip 2004, Tampere, Finland, Nov. 2004.

[Dally87] W. J. Dally and C. L. Seitz, "Deadlock free message routing in multiprocessor interconnection networks," IEEE Transactions on Computers. C-36, 5, pp. 547-553, May 1987.

[Dally98] W. J. Dalyy and J. W. Poulton, "Digital System Engineering," Cambridge University Press, 1989.

[Dally01] W. Dally and B. Towels, "Route packets, not wires: on-chip interconnection networks," DAC, Proceedings of the 38th Design Automation conference, pp.684-689, 2001.

[Dally02] W. Dally, "Computer Architecture is All About Interconnect," in Proceedings of the IEEE International Symposium on High-Performance Computer Architecture, February 2002.

[Dike99] C. Dike and E. T. Burton, "Miller and noise effect in a synchronizing flip-flop," in IEEE Journal of Solid-State Circuits, vol. 34, no. 6, pp. 849-855, June 1999.

[Dike00] C. Dike, "Synchronization Tutorial," presented at Sixth Int. Symposium on Advanced Research in Asynchronous Circuits and Systems (ASYNC2000), 2000.

[Dobkin04] R. R. Dobkin, R. Ginosar, and C. P. Sotiriou, "Data synchronization issues in GALS SoCs," in IEEE Proc. Int. Symp. on Asynchronous Circuits and Systems (ASYNC'04), 2004.

[Dobkin05] R. Dobkin, V. Vishnyakov, E. Friedman and R.Ginosar, "An Asynchronous Router for Multiple Service Levels Networks on Chip," in IEEE Proc. Int. Symp. on Asynchronous Circuits and Systems (ASYNC'05), pp.44-53, 2005

[Duato03] J. Duao, S. Yalamanchili and L. Ni, "Interconnection networks: an engineer approach," Morgan Kaufmann, San Francsco, CA, 2003.

[Edman04] A. Edman and C. Svensson, "Timing Closure through a Globally Synchronous, Timing Partitioned Design Methodology," Proc. of Design Automation Conference (DAC'04), pp. 71-74, 2004.

[Edman05] A. Edman, C. Svensson, and B. Mesgarzadeh, "Synchronous latency-insensitive design for multiple clock domain," SOC Conference, 2005. Proceedings. IEEE International, pp. 83-86, Sept. 2005.

[Elboim02] Y. Elboim, A. Kolodny, and R. Ginosar, "A clock tuning circuit for System-on-Chip," in Proc. 28th European Solid-State Circuits Conference (ESSCIRC 2002), 2002.

[Elboim03] Y. Elboim, A. Kolodny and R. Ginosar "A clock tuning circuit for system-on-chip," IEEE Trans. on Very Large Scale Integration (VLSI) Systems, v. 11 n. 4, pp. 616-626, August 2003.

[Gangw05] O. P. Gangwal, A. Radulescu, K. Goossens and S. Gonzalez Pestana, and E. Rijpkema, "Building Predictable Systems on Chip: An Analysis of Guaranteed Communication in the Æthereal Network on Chip," In Peter van der Stok, editor, Dynamic and Robust Streaming In and Between Connected Consumer-Electronics Devices. Springer, 2005.

[Ginosar03] R. Ginosar, "Fourteen ways to fool your synchronizer," in Proc. 9th IEEE Int. Symp. on Asynchronous Circuits and Systems (ASYNC'03), pp. 89-97, 2003.

[Glass94] C. Glass and L. Ni, "The turn model for adaptive routing," Journal of the Association for Computing Machinery, pp. 874–902, Sep. 1994.

[Goos02] K. Goossens, J. van Meerbergen, A. Peeters and P. Wielage "Networks on Silicon: Combining Best-Effort and Guaranteed Services," Design Automation and Test in Europe (DATE'02), 2002.

[Goos05] K. Goossens, J. Dielissen, and A. Radulescu "The Æthereal network on chip: Concepts, architectures, and implementations," IEEE Design and Test of Computers, Vol 22, pp. 414-421, Sept-Oct 2005.

[Goos05b] K. Goossens, J. Dielissen, O. P. Gangwal, S. Gonzalez Pestana, A. Radulescu, and E. Rijpkema "A Design Flow for Application-Specific Networks on Chip with Guaranteed Performance to Accelerate SOC Design and Verification," Proc. of Design, Automation and Test Conference in Europe (DATE05), March 2005.

[Green93] M. R. Greenstreet, "STARI: A Technique for High-Bandwidth Communication," PhD thesis, Department of Computer Science, Princeton University, Jan. 1993.

[Green95] M. R. Greenstreet, "Implementing a STARI chip," in Proc. of the 1995 Int. Conf. on Computer Design, pages 38-43, Austin, Texas, Oct. 1995.

[Guer00] P. Guerrier and A. Greiner, "A generic architecture for on-chip packet-switched interconnections," Proc. Design Automation and Test in Europe (DATE'00), pp. 250-256, Mars 2000.

[Guz07] Z. Guz, I. Walter, E. Bolotin, I. Cidon, R. Ginosar, and A. Kolodny, "Network Delays and Link Capacities in Application-Specific Wormhole NoCs," VLSI Design, vol. 2007, Article ID 90941, May 2007

[Hans07] A. Hansson, M. Coenen, and K. Goossens, "Channel trees: Reducing latency by sharing time slots in time-multiplexed networks on chip," Int'l Conf. on Hardware/Software Codesign and System Synthesis (CODES+ISSS), October 2007.

[ISO] www.iso.org

[ITRS05] International Technology Roadmap for Semiconductors. www.itrs.net, 2005.

[Jex95] J. Jex and C. Dike, "A fast resolving BiNMOS synchronizer for parallel processor interconnect, " IEEE Journal of Solid-State Circuits, Vol. 30, pp. 133 – 139, Feb 1995.

[Jex97] J. Jex, C. Dike and K. Self, "Fully asynchronous interface with programmable metastability settling time synchronizer," Patent 5,598,113, January 1997.

[Kaiser04] Stefan Kaiser et al., "4G MC-CDMA Multi Antenna System on Chip for Radio Enhancements (4MORE)," IST summit, Lyon, June 2004.

[Kessels05] Joep Kessels "Register-communication between mutually asynchronous domains," Proc. 11th IEEE Int. Symp. Asynchronous Circuits and Systems (ASYNC'05), 2005.

[Kinni02] D. J. Kinniment, A. Baystrov, and A. Yakovlev, "Synchronization circuit performances," IEEE Journal of Solid-State Circuits, vol. 37, pp. 202-209, 2002.

[Kim90] L. Kim and R. W. Dutton, "Metastability of CMOS latch/flip-flop," IEEE Journal of Solid State Circuits, vol. 25, no. 4, pp. 942-951, August 1990.

[Kol98] R. Kol and R. Ginosar, "Adaptive synchronization for multi-synchronous systems," in IEEE Int. Conf. Computer Design (ICCD'98), pp. 188-189, Oct. 1998.

[Leiser85] C. Leiserson, "Fat-Trees: Universal Networks for Hardware-Efficient Supercomputing," IEEE Trans. on Computers, vol. C-34, no. 10, pp. 892-901, October 1985

[Mauri03] P. Maurine, J.B. Rigaud, F. Bouesse, G. Sicard, and M. Renaudin, "Static Implementation of QDI Asynchronous Primitives," 13th International Workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS 2003), Torino, Italy, pp. 181-191, Sept. 2003.

[Mesga04] B. Mesgarzadeh, C. Svensson, and A. Alvandpour, "A new mesochronous clocking scheme for synchronization in SoC," in IEEE Int. Symp. on Circuits and Systems, May 2004.

[Miche06] G. De Micheli and L. Benini, "Networks on Chips," Morgan Kaufmann, 2006.

[Millbe04] M. Millberg, E. Nilsson, R. Thid and A. Jantsch "Guaranteed bandwidth using looped containers in temporally disjoint networks within the Nostrum network on chip," IEEE Proc. Design Automation and Test in Europe (DATE'04), vol. 2, pp. 890 - 895, Febrary 2004.

[Miro06] I. Miro-Panades, A. Greiner, and A. Sheibanyrad, "A low cost Network-on-Chip with guaranteed service well suited to the GALS approach," in IEEE 1st Int. Conf. on Nano-Networks, 2006.

[Miro07a] I. Miro-Panades, A. Greiner, "Bi-Synchronous FIFO for Synchronous Circuit Communication Well Suited for Network-on-Chip in GALS Architectures," First Inter. Symposium on Network-on-Chip (NOCS'07), pp. 83-94, May 2007.

[Miro07b] I. Miro-Panades, "Buffer memory control device (Dispositif de commnade d'une memoire tampon)," Patent FR2899985, October 2007.

[Miro08] I. Miro-Panades, "Control circuit for FIFO memory," Patent pending.

[Moore00] S.W. Moore, G.S. Taylor, P. A. Cunningham, R.D. Mullins, and P. Robinson, "Self calibrating clocks for globally asynchronous locally synchronous systems," in IEEE Proc. Int. Conf. on Computer Design, 2000.

[Mu01] F. Mu and C. Svensson, "Self-tested self-synchronization circuit for mesochronous clocking," in IEEE Transactions on Circuits and Systems-II, vol. 48, no. 2, pp. 129-140, Febr. 2001.

[Mutter99] J. Muttersbach, T. Villiger, K. Kaeslin, N. Felber and W. Fichtner, "Globally-Asynchronous Locally-Synchronous Architectures to Simplify the Design of On-CHIP Systems," Proc. 12th International ASIC/SOC Conference, pp. 317-321, Sept. 1999.

[Ni93] L.M. Ni and P.K. McKinley "A survey of wormhole routing techniques in direct networks," IEEE Computer 2 (1993) 62-75.

[NOSTR] http://www.imit.kth.se/info/FOFU/Nostrum/

[Pham98] G. N. Pham and K. C. Schmitt, "A high throughput, asynchronous, dual port FIFO memory implemented in ASIC technology," in Proc. Annual IEEE Int. ASIC Conf. and Exhibition, 1989.

[Qing00] W. Qing, M. Pedram and X. Wu "Clock-gating and its application to low power design of sequential circuits," IEEE Trans. Circuits Syst. I, Fundam. Theory Applicat., vol. 47, no3, pp.414-420, Mars 2000.

[Radu05] A. Radulescu, J. Dielissen, S. Gonzalez Pestana, Om Gangwal, E. Rijpkema, P. Wielage, and K. Goossens "An Efficient On-Chip Network Interface Offering Guaranteed Services, Shared-Memory Abstraction, and Flexible Network Programming," IEEE Trans. on CAD of Integrated Circuits and Systems, 24(1), January 2005.

[Rijpke03] E. Rijpkema, K. Goossens, A. Radulescu, J. Dielssen, J. van Meerbergen, P. Wielage and E. Waterlanden "Trade-offs in the design of a router with both guaranteed and best-effort services for networks on chip," IEEE. Proc.-Comput. Digit. Tech., Vol. 150, No 5, September 2003.

[Rougn04] N. Rougnon Glasson, "Device for transferring data between two asynchronous subsystems having a buffer memory," Patent US2004230723, Nov. 2004.

[Ru06] X. Ru, J. Dielissen, C. Svensson, K. Goossens "Synchronous Latency-Insensitive Design in Æthereal NoC," In Future Interconnects and Network on Chip workshop at Design, Automation and Test in Europe Conference and Exhibition (DATE), March 2006.

[Sarme95] L.F.G. Sarmenta, G.A. Pratt, and S.A. Ward, "Rational clocking," in Proc. ICCD, pp. 217-228, 1995.

[Sheiban06] A. Sheibanyrad and A. Greiner, "Two efficient synchronous<->asynchronous converters well-suited for network on chip in GALS architectures," in Int. workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS 2006), pp. 191-202, 2006.

[Sheiban07] A. Sheibanyrad, I. Miro-Panades, and A. Greiner, "Systematic comparison between the asynchronous and the multi-synchronous implementations of a Network on Chip architecture," in Proc. IEEE Design, Automation and Test in Europe (DATE'07), April 2007.

[Sjogren00] A. E. Sjogren and C. J. Myers, "Interfacing synchronous and asynchronous modules within a high-speed pipeline," in IEEE Trans. VLSI Syst., Vol 8, no. 5, pp 573-583, Oct. 2000.

[SOCLIB] http://www.soclib.fr

[Steenh06] F. Steenhof, H. Duque, B. Nilsson, K. Goossens, and R. P. Llopis, "Networks on Chips for High-End Consumer-Electronics TV System Architectures," In Proc. of Design, Automation and Test Conference in Europe (DATE'06), March 2006.

[Svenss04] C. Svensson, "Synchronous latency insensitive design," Int. Symp. On Asynchronous Circuits and Systems (ASYNC'04), 19-23 April 2004.

[Tamir03] Guy Tamir, "Synchronization metastability," Research thesis, Technion – Israel, April 2003.

[Vang05] S. Vangal, N. Y. Borkar, and A. Alvandpour, "A Six-Port 57GB/s Double-Pumped Non-blocking Router Core," Dig. Symp. VLSI Circuits, pp. 268-269, June 2005.

[Vang06] S. Vangal, Y. Hoskote, N. Y. Borkar, et al., "A 6.2-GFlops Floating Point Multiply-Accumulator with Conditional Normalization," IEEE J. Solid-State Circuits, pp. 2314-2323, Oct. 2006.

[Vang07] S. Vangal et al., "An 80-Tile 1.28TFLOPS Network-on-Chip in 65nm CMOS," ISSCC Dig. Tech. Papers, pp. 98-99, Feb. 2007.

[Vang07b] S. Vanlgal et al., "A 5.1GHz 0.34mm² Router for Network-on-Chip Applications," IEEE Symposium on VLSI Circuits, pp. 42-43, June 2007.

[VCI00] VSI Alliance. Virtual Component Interface Standard (OCB2 2.0), August 2000. http://www.vsia.com/

[Wilso01] H. Wilson and M. Haycock., "A six-port 30-GB/s non-blocking router component using point-to-point simultaneous bidirectional signaling for high-bandwidth interconnects," IEEE JSSC, vol. 36, pp. 1954–1963, Dec. 2001.

[Yu06] Z. Yu, M. Meeuwsen, R. Apperson, O. Sattari, M. Lai, J. Webb, E. Work, T. Mohsenin, M. Singh, and B. Baas, "An Asynchronous Array of Simple Processors for DSP Applications," Proc. of the IEEE International Solid-State Circuits Conference (ISSCC '06), pp. 428-429, February 2006.

[Yu07] Zhiyi Yu, "High Performance and Energy Efficient Multi-core Systems for DSP Applications," Ph.D Dissertation, Technical Report ECE-CE-2007-5, Computer Engineering Research Laboratory, ECE Department, University of California, Davis, 2007.

[Zipf04] P. Zipf, H. Hinkelmann, A. Ashraf, and M. Glesner, "A switch architecture and signal synchronization for GALS System-on-Chip," in Proc. of 17th Symposium on Integrated Circuits and Systems Design (SBCCI2004), pages 210-215, 2004.